NLP Project 1 : Review Project Analysis.

DESCRIPTION

Help a leading mobile brand understand the voice of the customer by analyzing the reviews of their product on Amazon and the topics that customers are talking about. You will perform topic modeling on specific parts of speech. You'll finally interpret the emerging topics.

Problem Statement:

A popular mobile phone brand, Lenovo has launched their budget smartphone in the Indian market. The client wants to understand the VOC (voice of the customer) on the product. This will be useful to not just evaluate the current product, but to also get some direction for developing the product pipeline. The client is particularly interested in the different aspects that customers care about. Product reviews by customers on a leading e-commerce site should provide a good view.

Domain: Amazon reviews for a leading phone brand

Analysis to be done: POS tagging, topic modeling using LDA, and topic interpretation

Content:

Dataset: 'K8 Reviews v0.2.csv'

Columns:

1. Sentiment: The sentiment against the review (4,5 star reviews are positive, 1,2 are negative)
2. Reviews: The main text of the review

Steps to perform:

Discover the topics in the reviews and present it to business in a consumable format. Employ techniques in syntactic processing and topic modeling.

Perform specific cleanup, POS tagging, and restricting to relevant POS tags, then, perform topic modeling using LDA. Finally, give business-friendly names to the topics and make a table for business.

Tasks:

1. Read the .csv file using Pandas. Take a look at the top few records.
2. Normalize casings for the review text and extract the text into a list for easier manipulation.
3. Tokenize the reviews using NLTKs word_tokenize function.
4. Perform parts-of-speech tagging on each sentence using the NLTK POS tagger.
5. For the topic model, we should want to include only nouns.
    A. Find out all the POS tags that correspond to nouns.
    B. Limit the data to only terms with these tags.
6. Lemmatize.
    A. Different forms of the terms need to be treated as one.
    B. No need to provide POS tag to lemmatizer for now.
7. Remove stopwords and punctuation (if there are any).
8. Create a topic model using LDA on the cleaned-up data with 12 topics.
    A. Print out the top terms for each topic.
    B. What is the coherence of the model with the c_v metric?
9. Analyze the topics through the business lens.
    A. Determine which of the topics can be combined.
10. Create topic model using LDA with what you think is the optimal number of topics
    A. What is the coherence of the model?
11. The business should be able to interpret the topics.
    A. Name each of the identified topics.
    B. Create a table with the topic name and the top 10 terms in each to present to the business.

Import the necessary libraries

In [1]:
```
#Let's import Numpy and pandas for dataframes
import numpy as np
import pandas as pd

#Let's import nltk library for tokenization, lemmatization, stopwords, pos tags and FreqDist
import nltk
from nltk.tokenize import word_tokenize,TweetTokenizer
from nltk.stem import WordNetLemmatizer
from nltk.tag import pos_tag
from nltk.corpus import stopwords
from nltk import FreqDist

#Let's import string library for punctuation and string manipulations
import string

#Let's import Gensim library for LDA model creation,Corpora in gensim to create the id2word Dictionary and corpus of terms
import gensim
import gensim.corpora as corpora
from gensim.models import CoherenceModel

#Let's import matplotlib and pyLDAvis for the LDA model visualization
import matplotlib.pyplot as plt
import pyLDAvis
import pyLDAvis.gensim_models

#Let's import warnings to ignore deprecation warnings
import warnings
warnings.filterwarnings('ignore')
```

1. Read the .csv file using Pandas. Take a look at the top few records.

In [2]:
```
#Let's load the dataset into the environment
reviews_data = pd.read_csv('K8_Reviews_v0.2.csv')

#Let's look at the top 5 records from the dataset
reviews_data.head()
```

Out[2]:

	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...

2. Normalize casings for the review text and extract the text into a list for easier manipulation.

In [3]:
```
#Let's Normalize the text by reducing to lower case and convert the text into a list
review_list = [review.lower() for review in reviews_data['review']]
print(review_list[:5])
```

```
['good but need updates and improvements', "worst mobile i have bought ever, battery is draining like hell, backup is only 6 to 7 hours with internet uses, even if i put m
ile idle its getting discharged.this is biggest lie from amazon & lenove which is not at all expected, they are making full by saying that battery is 4000mah & booster cha
er is fake, it takes at least 4 to 5 hours to be fully charged.don't know how lenovo will survive by making full of us.please don;t go for this else you will regret like m
e.", '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 still same and the a
zon is not returning the phone .highly disappointing of amazon']
```

3. Tokenize the reviews using NLTKs word_tokenize function.

In [4]:
```
#Let's Tokenize the reviews
review_words = [word_tokenize(review) for review in review_list]
print(review_words[:5])
```

```
[['good', 'but', 'need', 'updates', 'and', 'improvements'], ['worst', 'mobile', 'i', 'have', 'bought', 'ever', ',', 'battery', 'is', 'draining', 'like', 'hell', ',', 'back
p', 'is', 'only', '6', 'to', '7', 'hours', 'with', 'internet', 'uses', ',', 'even', 'if', 'i', 'put', 'mobile', 'idle', 'its', 'getting', 'discharged.this', 'is', 'biggest
'lie', 'from', 'amazon', '&', 'lenove', 'which', 'is', 'not', 'at', 'all', 'expected', ',', 'they', 'are', 'making', 'full', 'by', 'saying', 'that', 'battery', 'is', '4000
h', '&', 'booster', 'charger', 'is', 'fake', ',', 'it', 'takes', 'at', 'least', '4', 'to', '5', 'hours', 'to', 'be', 'fully', 'charged.do', "n't", 'know', 'how', 'lenovo',
'will', 'survive', 'by', 'making', 'full', 'of', 'us.please', 'don', ';', 't', 'go', 'for', 'this', 'else', 'you', 'will', 'regret', 'like', 'me', '.'], ['when', 'i', 'wil
l', 'get', 'my', '10', '%', 'cash', 'back', '....', 'its', 'already', '15', 'january', '..'], ['good'], ['the', 'worst', 'phone', 'everthey', 'have', 'changed', 'the', 'la
t', 'phone', 'but', 'the', 'problem', 'is', 'still', 'same', 'and', 'the', 'amazon', 'is', 'not', 'returning', 'the', 'phone', '.highly', 'disappointing', 'of', 'amazon']]
```

4. Perform parts-of-speech tagging on each sentence using the NLTK POS tagger.

In [5]:
```
#Let's perform POS tagging using NLTK pos tagger
pos_tagged_review = [pos_tag(review) for review in review_words]
print('Length of POS tags = ',len(pos_tagged_review))
print(pos_tagged_review[:5])
```

```
Length of POS tags =  14675
[[('good', 'JJ'), ('but', 'CC'), ('need', 'VBP'), ('updates', 'NNS'), ('and', 'CC'), ('improvements', 'NNS')], [('worst', 'JJS'), ('mobile', 'NN'), ('i', 'NN'), ('have', 'V
P'), ('bought', 'VBN'), ('ever', 'RB'), (',', ','), ('battery', 'NN'), ('is', 'VBZ'), ('draining', 'VBG'), ('like', 'IN'), ('hell', 'NN'), (',', ','), ('backup', 'NN'), ('
s', 'VBZ'), ('only', 'RB'), ('6', 'CD'), ('to', 'TO'), ('7', 'CD'), ('hours', 'NNS'), ('with', 'IN'), ('internet', 'JJ'), ('uses', 'NNS'), (',', ','), ('even', 'RB'), ('if
'IN'), ('i', 'JJ'), ('put', 'VBP'), ('mobile', 'JJ'), ('idle', 'NN'), ('its', 'PRP$'), ('getting', 'VBG'), ('discharged.this', 'NN'), ('is', 'VBZ'), ('biggest', 'JJS'), ('
e', 'NN'), ('from', 'IN'), ('amazon', 'NN'), ('&', 'CC'), ('lenove', 'NN'), ('which', 'WDT'), ('is', 'VBZ'), ('not', 'RB'), ('at', 'IN'), ('all', 'DT'), ('expected', 'VBN'
(',', ','), ('they', 'PRP'), ('are', 'VBP'), ('making', 'VBG'), ('full', 'JJ'), ('by', 'IN'), ('saying', 'VBG'), ('that', 'DT'), ('battery', 'NN'), ('is', 'VBZ'), ('4000ma
h', 'CD'), ('&', 'CC'), ('booster', 'JJR'), ('charger', 'NN'), ('is', 'VBZ'), ('fake', 'JJ'), (',', ','), ('it', 'PRP'), ('takes', 'VBZ'), ('at', 'IN'), ('least', 'JJS'),
('4', 'CD'), ('to', 'TO'), ('5', 'CD'), ('hours', 'NNS'), ('to', 'TO'), ('be', 'VB'), ('fully', 'RB'), ('charged.do', 'VBP'), ("n't", 'RB'), ('know', 'VB'), ('how', 'WRB')
('lenovo', 'JJ'), ('will', 'MD'), ('survive', 'VB'), ('by', 'IN'), ('making', 'VBG'), ('full', 'JJ'), ('of', 'IN'), ('us.please', 'JJ'), ('don', 'NN'), (';', ':'), ('t', '
C'), ('go', 'VB'), ('for', 'IN'), ('this', 'DT'), ('else', 'JJ'), ('you', 'PRP'), ('will', 'MD'), ('regret', 'VB'), ('like', 'IN'), ('me', 'PRP'), ('.', '.')], [('when', '
B'), ('i', 'NN'), ('will', 'MD'), ('get', 'VB'), ('my', 'PRP$'), ('10', 'CD'), ('%', 'NN'), ('cash', 'NN'), ('back', 'RB'), ('....', 'VBZ'), ('its', 'PRP$'), ('already', '
B'), ('15', 'CD'), ('january', 'JJ'), ('..', 'NN')], [('good', 'JJ')], [('the', 'DT'), ('worst', 'JJS'), ('phone', 'NN'), ('everthey', 'NN'), ('have', 'VBP'), ('changed',
BN'), ('the', 'DT'), ('last', 'JJ'), ('phone', 'NN'), ('but', 'CC'), ('the', 'DT'), ('problem', 'NN'), ('is', 'VBZ'), ('still', 'RB'), ('same', 'JJ'), ('and', 'CC'), ('the
'DT'), ('amazon', 'NN'), ('is', 'VBZ'), ('not', 'RB'), ('returning', 'VBG'), ('the', 'DT'), ('phone', 'NN'), ('.highly', 'RB'), ('disappointing', 'JJ'), ('of', 'IN'), ('am
on', 'NN')]]
```

5. For the topic model, we should want to include only nouns.

- Find out all the POS tags that correspond to nouns.
- Limit the data to only terms with these tags.

In [6]:
```
#Let's find out all the POS tags that correspond to nouns and limit the data to only terms with noun tags

pos_noun_reviews = [] #Creating a empty list
for review in pos_tagged_review:
    nouns=[]
    for word in review:
        if "NN" in word[1]:
            nouns.append(word)
    pos_noun_reviews.append(nouns)
```

In [7]:
```
print('Length :',len(pos_noun_reviews))
```

```
Length : 14675
```

In [8]:
```
print(pos_noun_reviews[:30])
```

```
[[('updates', 'NNS'), ('improvements', 'NNS')], [('mobile', 'NN'), ('i', 'NN'), ('battery', 'NN'), ('hell', 'NN'), ('backup', 'NN'), ('hours', 'NNS'), ('uses', 'NNS'), ('i
e', 'NN'), ('discharged.this', 'NN'), ('lie', 'NN'), ('amazon', 'NN'), ('lenove', 'NN'), ('battery', 'NN'), ('charger', 'NN'), ('hours', 'NNS'), ('don', 'NN')], [('i', 'N
N'), ('%', 'NN'), ('cash', 'NN'), ('..', 'NN')], [], [('phone', 'NN'), ('everthey', 'NN'), ('phone', 'NN'), ('problem', 'NN'), ('amazon', 'NN'), ('phone', 'NN'), ('amazon'
'NN')], [('camerawaste', 'NN'), ('money', 'NN')], [('phone', 'NN'), ('allot', 'NN'), ('..', 'NNP'), ('reason', 'NN'), ('k8', 'NNS')], [('battery', 'NN'), ('level', 'NN')],
[('problems', 'NNS'), ('phone', 'NN'), ('hanging', 'NN'), ('problems', 'NNS'), ('note', 'NN'), ('station', 'NN'), ('ahmedabad', 'NN'), ('years', 'NNS'), ('phone', 'NN'), (
enovo', 'NN')], [('lot', 'NN'), ('glitches', 'NNS'), ('thing', 'NN'), ('options', 'NNS')], [('wrost', 'NN')], [('phone', 'NN'), ('charger', 'NN'), ('damage', 'NN'), ('mont
s', 'NNS')], [('item', 'NN'), ('battery', 'NN'), ('life', 'NN')], [('i', 'NNS'), ('battery', 'NN'), ('problem', 'NN'), ('motherboard', 'NN'), ('problem', 'NN'), ('months',
'NNS'), ('mobile', 'NN'), ('life', 'NN')], [('phone', 'NN'), ('slim', 'NN'), ('battry', 'NN'), ('backup', 'NN'), ('screen', 'NN')], [('headset', 'NN')], [('time', 'NN'),
('i', 'NN')], [('product', 'NN'), ('prize', 'NN'), ('range', 'NN'), ('specification', 'NN'), ('comparison', 'NN'), ('mobile', 'NN'), ('range', 'NN'), ('i', 'NN'), ('phone'
'NN'), ('seal', 'NN'), ('i', 'NNS'), ('credit', 'NN'), ('card', 'NN'), ('i', 'NN'), ('..', 'NNP'), ('..', 'NNP'), ('deal', 'NN'), ('amazon', 'NN'), ('..', 'NN')], [('batte
y', 'NN'), ('..', 'NNP'), ('solutions', 'NNS'), ('battery', 'NN'), ('life', 'NN')], [('smartphone', 'NN')], [], [('galery', 'NN'), ('problem', 'NN'), ('speaker', 'NN'), ('
one', 'NN')], [('camera', 'NN'), ('speed.excellent', 'NN'), ('features.excelent', 'NN'), ('battery', 'NN')], [('product', 'NN')], [('product', 'NN'), ('camera', 'NN'), ('o
s', 'NN'), ('battery', 'NN'), ('phone', 'NN'), ('product', 'NN'), ('..', 'NN')], [('options', 'NNS'), ('cast', 'NN'), ('screen', 'NN'), ('wifi', 'NN'), ('call', 'NN'), ('o
ion', 'NN'), ('mobile', 'NN'), ('hotspot', 'NN')], [('phone', 'NN'), ('usb', 'NN'), ('cable', 'NN')], [('phone', 'NN'), ('price', 'NN'), ('mobile', 'NN'), ('lenovo', 'NN')
('display', 'NN')], [('specifications', 'NNS'), ('functions', 'NNS'), ('phone', 'NN'), ('any1', 'NN'), ('..', 'NNS')], [('i', 'NN'), ('fon', 'NN'), ('fon', 'NN'), ('speeka
s', 'NNS'), ('i', 'NN')]]
```

In [9]:
```
# Let's Exclude the reviews that did not have any nouns (In case if the reviews are blank or empty)

pos_noun_reviews=[review for review in pos_noun_reviews if len(review)>=1]
```

In [10]:
```
print('Length : ',len(pos_noun_reviews))
```

```
Length :  13487
```

In [11]:
```
print(pos_noun_reviews[:30])
```

```
[[('updates', 'NNS'), ('improvements', 'NNS')], [('mobile', 'NN'), ('i', 'NN'), ('battery', 'NN'), ('hell', 'NN'), ('backup', 'NN'), ('hours', 'NNS'), ('uses', 'NNS'), ('i
e', 'NN'), ('discharged.this', 'NN'), ('lie', 'NN'), ('amazon', 'NN'), ('lenove', 'NN'), ('battery', 'NN'), ('charger', 'NN'), ('hours', 'NNS'), ('don', 'NN')], [('i', 'N
N'), ('%', 'NN'), ('cash', 'NN'), ('..', 'NN')], [('phone', 'NN'), ('everthey', 'NN'), ('phone', 'NN'), ('problem', 'NN'), ('amazon', 'NN'), ('phone', 'NN'), ('amazon', 'N
N')], [('camerawaste', 'NN'), ('money', 'NN')], [('phone', 'NN'), ('allot', 'NN'), ('..', 'NNP'), ('reason', 'NN'), ('k8', 'NNS')], [('battery', 'NN'), ('level', 'NN')],
[('problems', 'NNS'), ('phone', 'NN'), ('hanging', 'NN'), ('problems', 'NNS'), ('note', 'NN'), ('station', 'NN'), ('ahmedabad', 'NN'), ('years', 'NNS'), ('phone', 'NN'), (
enovo', 'NN')], [('lot', 'NN'), ('glitches', 'NNS'), ('thing', 'NN'), ('options', 'NNS')], [('wrost', 'NN')], [('phone', 'NN'), ('charger', 'NN'), ('damage', 'NN'), ('mont
s', 'NNS')], [('item', 'NN'), ('battery', 'NN'), ('life', 'NN')], [('i', 'NNS'), ('battery', 'NN'), ('problem', 'NN'), ('motherboard', 'NN'), ('problem', 'NN'), ('months',
'NNS'), ('mobile', 'NN'), ('life', 'NN')], [('phone', 'NN'), ('slim', 'NN'), ('battry', 'NN'), ('backup', 'NN'), ('screen', 'NN')], [('headset', 'NN')], [('time', 'NN'),
('i', 'NN')], [('product', 'NN'), ('prize', 'NN'), ('range', 'NN'), ('specification', 'NN'), ('comparison', 'NN'), ('mobile', 'NN'), ('range', 'NN'), ('i', 'NN'), ('phone'
'NN'), ('seal', 'NN'), ('i', 'NNS'), ('credit', 'NN'), ('card', 'NN'), ('i', 'NN'), ('..', 'NNP'), ('..', 'NNP'), ('deal', 'NN'), ('amazon', 'NN'), ('..', 'NN')], [('batte
y', 'NN'), ('..', 'NNP'), ('solutions', 'NNS'), ('battery', 'NN'), ('life', 'NN')], [('smartphone', 'NN')], [('galery', 'NN'), ('problem', 'NN'), ('speaker', 'NN'), ('phon
e', 'NN')], [('camera', 'NN'), ('speed.excellent', 'NN'), ('features.excelent', 'NN'), ('battery', 'NN')], [('product', 'NN')], [('product', 'NN'), ('camera', 'NN'), ('os'
'NN'), ('battery', 'NN'), ('phone', 'NN'), ('product', 'NN'), ('..', 'NN')], [('options', 'NNS'), ('cast', 'NN'), ('screen', 'NN'), ('wifi', 'NN'), ('call', 'NN'), ('optio
n', 'NN'), ('mobile', 'NN'), ('hotspot', 'NN')], [('phone', 'NN'), ('usb', 'NN'), ('cable', 'NN')], [('phone', 'NN'), ('price', 'NN'), ('mobile', 'NN'), ('lenovo', 'NN'),
('display', 'NN')], [('specifications', 'NNS'), ('functions', 'NNS'), ('phone', 'NN'), ('any1', 'NN'), ('..', 'NNS')], [('i', 'NN'), ('fon', 'NN'), ('fon', 'NN'), ('speeka
s', 'NNS'), ('i', 'NN')], [('phone', 'NN'), ('issue', 'NN'), ('colors', 'NNS'), ('screen', 'NN')], [('update', 'NN'), ('oreo', 'NN'), ('battery', 'NN'), ('back-up', 'NN'),
('heating', 'NN'), ('problem', 'NN'), ('update', 'NN'), ('phone', 'NN'), ('battery', 'NN'), ('update', 'NN'), ('oreo', 'NN'), ('😒😒☹', 'NN')]]
```

6. Lemmatize.

- Different forms of the terms need to be treated as one.

- No need to provide POS tag to lemmatizer for now.

In [12]:
```
# Let's Lemmatize different forms of the nouns without including POS tags into lemmatizer

wnl = WordNetLemmatizer()
lemmatized_words =[]
for review in pos_noun_reviews:
    lemma_word=[]
    for word in review:
        lemma_word.append(wnl.lemmatize(word[0]))
    lemmatized_words.append(lemma_word)
```

In [13]:
```
print(lemmatized_words[:30])
```

```
[['update', 'improvement'], ['mobile', 'i', 'battery', 'hell', 'backup', 'hour', 'us', 'idle', 'discharged.this', 'lie', 'amazon', 'lenove', 'battery', 'charger', 'hour',
on'], ['i', '%', 'cash', '..'], ['phone', 'everthey', 'phone', 'problem', 'amazon', 'phone', 'amazon'], ['camerawaste', 'money'], ['phone', 'allot', '..', 'reason', 'k8'],
['battery', 'level'], ['problem', 'phone', 'hanging', 'problem', 'note', 'station', 'ahmedabad', 'year', 'phone', 'lenovo'], ['lot', 'glitch', 'thing', 'option'], ['wrost'
['phone', 'charger', 'damage', 'month'], ['item', 'battery', 'life'], ['i', 'battery', 'problem', 'motherboard', 'problem', 'month', 'mobile', 'life'], ['phone', 'slim', '
ttry', 'backup', 'screen'], ['headset'], ['time', 'i'], ['product', 'prize', 'range', 'specification', 'comparison', 'mobile', 'range', 'i', 'phone', 'seal', 'i', 'credit'
'card', 'i', '..', '..', 'deal', 'amazon', '..'], ['battery', '..', 'solution', 'battery', 'life'], ['smartphone'], ['galery', 'problem', 'speaker', 'phone'], ['camera', '
eed.excellent', 'features.excelent', 'battery'], ['product'], ['product', 'camera', 'o', 'battery', 'phone', 'product', '..'], ['option', 'cast', 'screen', 'wifi', 'call',
'option', 'mobile', 'hotspot'], ['phone', 'usb', 'cable'], ['phone', 'price', 'mobile', 'lenovo', 'display'], ['specification', 'function', 'phone', 'any1', '..'], ['i', '
n', 'fon', 'speekars', 'i'], ['phone', 'issue', 'color', 'screen'], ['update', 'oreo', 'battery', 'back-up', 'heating', 'problem', 'update', 'phone', 'battery', 'update',
reo', '😒😒☹']]
```

7. Remove stopwords and punctuation (if there are any).

In [14]:
```
# The o/p from the above lemmatizer still has many composite words,contains emojis,special characters etc.
# So Lets make use of TweetTokenizer() that helps to tokenize Tweet Corpus into relevant tokens.
# The advantage of using TweetTokenizer() compared to regular word_tokenize is that, when processing tweets,
# we often come across emojis, hashtags that need to be handled differently.

tweet_tokenize = TweetTokenizer()

#Let's Create a list of stopwords with punctuations& Manually added token ['\s'] as this is usually seperated in tokenize
stop_words = stopwords.words("english")
stop_words = stop_words+list(string.punctuation)+["\'s"]

filtered_review_words=[]

for review in lemmatized_words:
    filter_words=[]
    for words in review:
        review_words = []
        review_words = tweet_tokenize.tokenize(words)
        for word in review_words:
            if word not in stop_words:
                filter_words.append(word)
    filtered_review_words.append(filter_words)
```

In [15]:
```
print('Length :',len(filtered_review_words))
```

```
Length : 13487
```

In [16]:
```
#Let's Exclude the reviews that contains only stopwords as these reviews might be blank or empty
filtered_review_words=[review for review in filtered_review_words if len(review)>=1]
```

In [17]:
```
print('Length :',len(filtered_review_words))
```

```
Length : 13453
```

In [18]:
```
print(filtered_review_words[:30])
```

```
[['update', 'improvement'], ['mobile', 'battery', 'hell', 'backup', 'hour', 'us', 'idle', 'discharged.this', 'lie', 'amazon', 'lenove', 'battery', 'charger', 'hour'], ['ca
h', '..'], ['phone', 'everthey', 'phone', 'problem', 'amazon', 'phone', 'amazon'], ['camerawaste', 'money'], ['phone', 'allot', '..', 'reason', 'k8'], ['battery', 'level']
['problem', 'phone', 'hanging', 'problem', 'note', 'station', 'ahmedabad', 'year', 'phone', 'lenovo'], ['lot', 'glitch', 'thing', 'option'], ['wrost'], ['phone', 'charger'
'damage', 'month'], ['item', 'battery', 'life'], ['battery', 'problem', 'motherboard', 'problem', 'month', 'mobile', 'life'], ['phone', 'slim', 'battry', 'backup', 'scree
n'], ['headset'], ['time'], ['product', 'prize', 'range', 'specification', 'comparison', 'mobile', 'range', 'phone', 'seal', 'credit', 'card', '..', '..', 'deal', 'amazon'
'..'], ['battery', '..', 'solution', 'battery', 'life'], ['smartphone'], ['galery', 'problem', 'speaker', 'phone'], ['camera', 'speed.excellent', 'features.excelent', 'bat
ry'], ['product'], ['product', 'camera', 'battery', 'phone', 'product', '..'], ['option', 'cast', 'screen', 'wifi', 'call', 'option', 'mobile', 'hotspot'], ['phone', 'usb'
'cable'], ['phone', 'price', 'mobile', 'lenovo', 'display'], ['specification', 'function', 'phone', '1', '..'], ['fon', 'fon', 'speekars'], ['phone', 'issue', 'color', 'sc
en'], ['update', 'oreo', 'battery', 'back-up', 'heating', 'problem', 'update', 'phone', 'battery', 'update', 'oreo', '😒', '😒', '☹', '']]
```

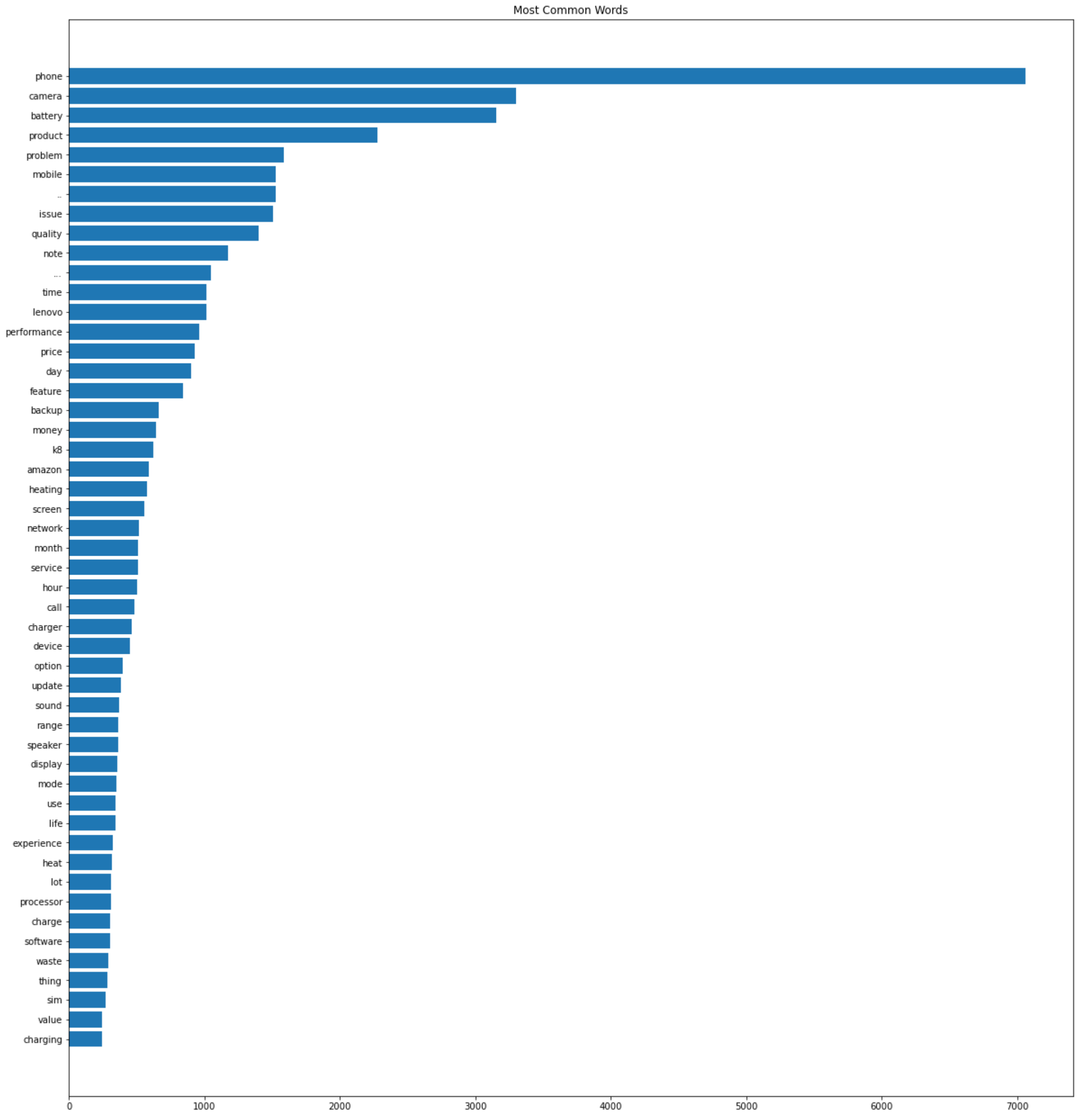
In [19]:

```
#Let's create a barplot to visualize the 50 most common words

list_of_words = [word for review in filtered_review_words for word in review]
common_word_freq=FreqDist(list_of_words).most_common(50)
word_list = common_word_freq[::-1]

words,freq = [],[]
for word in word_list:
    words.append(word[0])
    freq.append(word[1])
x=np.array(words)
y=np.array(freq)

plt.figure(figsize=(20,22))
plt.barh(x,y)
plt.title('Most Common Words')
plt.show()
```



In [20]:
```
print(common_word_freq)
```

```
[('phone', 7062), ('camera', 3303), ('battery', 3157), ('product', 2279), ('problem', 1589), ('mobile', 1530), ('..', 1527), ('issue', 1509), ('quality', 1404), ('note', 17), ('...', 1047), ('time', 1019), ('lenovo', 1013), ('performance', 961), ('price', 931), ('day', 905), ('feature', 844), ('backup', 667), ('money', 644), ('k8', 626), ('azon', 588), ('heating', 575), ('screen', 555), ('network', 519), ('month', 508), ('service', 508), ('hour', 506), ('call', 483), ('charger', 467), ('device', 449), ('option', 399), ('update', 385), ('sound', 369), ('range', 368), ('speaker', 366), ('display', 356), ('mode', 354), ('use', 344), ('life', 343), ('experience', 328), ('heat', 316), ('lot', 314), ('processor', 309), ('charge', 307), ('software', 303), ('waste', 289), ('thing', 282), ('sim', 270), ('value', 247), ('charging', 246)]
```

From the above o/p we see that there are some punctuations that are still appear in the word tokens like '..' and '...'. we can also see that the list consists of emojis which are not relevant for topic modelling,Further there are also just numbers in the place of words - like '1' and '2' which are not relevant. From the topic modelling perspective we can remove the obvious and contextual stop words like 'hi','hello','phone','lenovo','mobile','k8','product'.

In [21]:
```
#Let's Revise the stopwords based on the above analysis
aditional_stop_words= ["...","..",'phone','good','bad','lenovo','k8','note','product',
                       'mobile','hai','please','pls','star','hi','ho','ok','superb','handset']
stop_words = stop_words + aditional_stop_words

#isalnum() to remove emoji an isnumeric() to remove only number tokens present in the list
#len(word)!=1 will eliminate all one letter tokens such as 'u','i' etc.
final_review_words = []
for review in filtered_review_words:
    stopwords_removed_review=[]
    for word in review:
        if word not in stop_words and word.isalnum() and (not word.isnumeric()) and len(word)!=1:
            stopwords_removed_review.append(word)
    final_review_words.append(stopwords_removed_review)
```

In [22]:
```
print('Length :',len(final_review_words))
```

```
Length : 13453
```

In [23]:
```
#let's Clear any reviews which are now empty lists after the removal of revised stop words
final_review_words=[review for review in final_review_words if len(review)>=1]
```

In [24]:
```
print('Length :',len(final_review_words))
```

```
Length : 11858
```

In [25]:
```
print(final_review_words[:50])
```

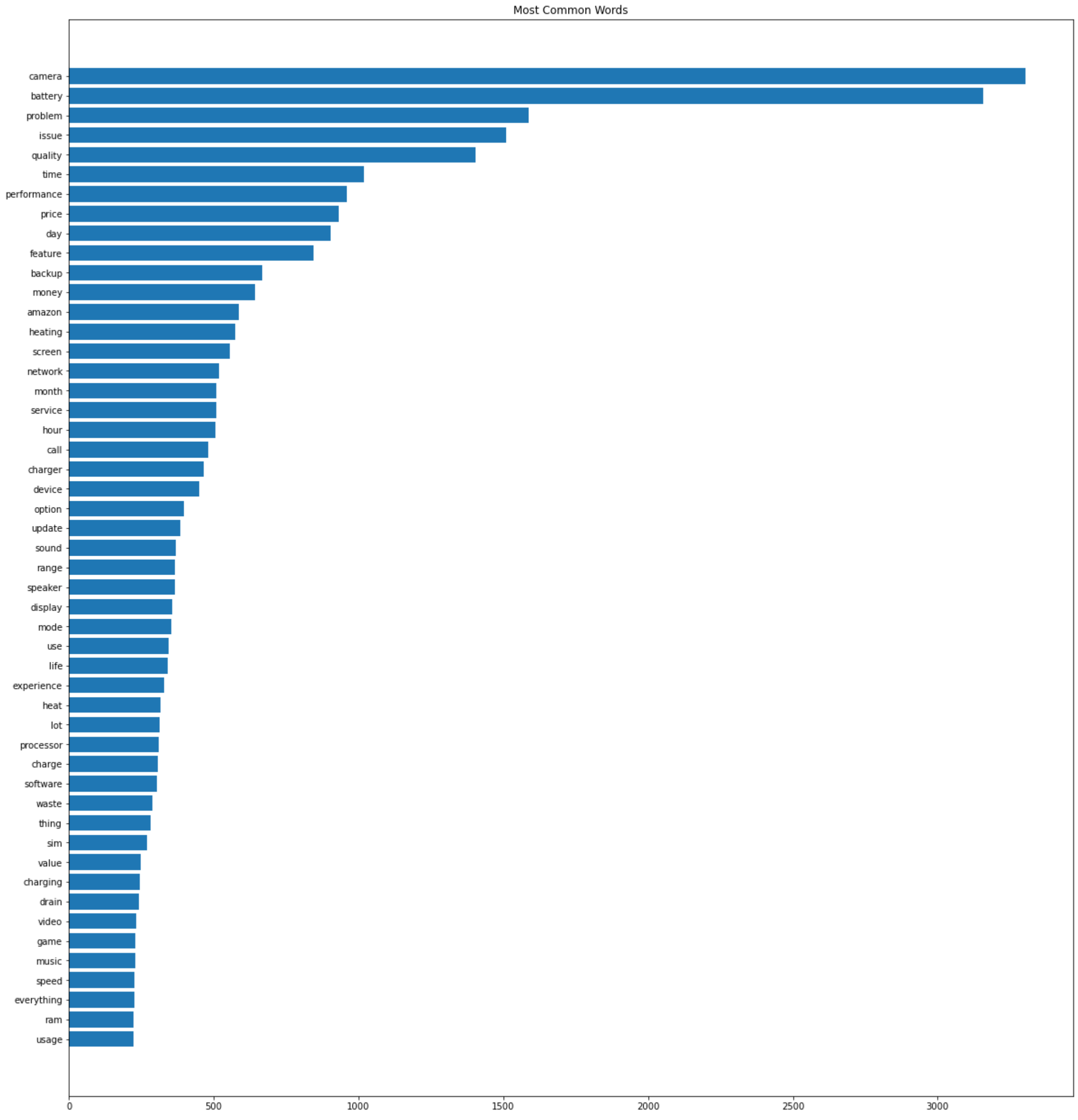
```
[['update', 'improvement'], ['battery', 'hell', 'backup', 'hour', 'us', 'idle', 'lie', 'amazon', 'lenove', 'battery', 'charger', 'hour'], ['cash'], ['everthey', 'problem', 'amazon', 'amazon'], ['camerawaste', 'money'], ['allot', 'reason'], ['battery', 'level'], ['problem', 'hanging', 'problem', 'station', 'ahmedabad', 'year'], ['lot', 'glitch', 'thing', 'option'], ['wrost'], ['charger', 'damage', 'month'], ['item', 'battery', 'life'], ['battery', 'problem', 'motherboard', 'problem', 'month', 'life'], ['slim', 'battry', 'backup', 'screen'], ['headset'], ['time'], ['prize', 'range', 'specification', 'comparison', 'range', 'seal', 'credit', 'card', 'deal', 'amazon'], ['battery', 'lution', 'battery', 'life'], ['smartphone'], ['galery', 'problem', 'speaker'], ['camera', 'battery'], ['camera', 'battery'], ['option', 'cast', 'screen', 'wifi', 'call', 'tion', 'hotspot'], ['usb', 'cable'], ['price', 'display'], ['specification', 'function'], ['fon', 'fon', 'speekars'], ['issue', 'color', 'screen'], ['update', 'oreo', 'batry', 'heating', 'problem', 'update', 'battery', 'update', 'oreo'], ['one', 'sim', 'customer', 'service'], ['performance', 'battery'], ['camera', 'backup', 'pricefull', 'paa', 'wasole'], ['performance', 'signal', 'restarts', 'bcoms', 'plzz', 'dont'], ['round', 'performance'], ['rs', 'span', 'day', 'trust', 'deal', 'amazon'], ['disappointment 'signal', 'problem', 'headache', 'problem', 'call', 'range'], ['rate', 'camera', 'quality'], ['price', 'feature'], ['price', 'quality', 'camera'], ['min', 'quality'], ['ise'], ['sell', 'take', 'hr', 'charge'], ['ekdam', 'network', 'problem', 'camera', 'quality', 'performance'], ['performance', 'rm', 'memory'], ['featurs', 'battery'], ['value', 'money'], ['problem', 'speaker', 'breakup', 'problem', 'hour'], ['heating', 'problem', 'choice'], ['battery', 'standby', 'problem'], ['battery', 'camera', 'jet', 'speed', 'apps']]
```

In [26]:
```
#Let's create a barplot to visualize the 50 most common words

list_of_words = [word for review in final_review_words for word in review]
word_freq=FreqDist(list_of_words).most_common(50)
word_list_2 = word_freq[::-1]

words,freq = [],[]
for word in word_list_2:
    words.append(word[0])
    freq.append(word[1])
x=np.array(words)
y=np.array(freq)

plt.figure(figsize=(20,22))
plt.barh(x,y)
plt.title('Most Common Words')
plt.show()
```



8. Create a topic model using LDA on the cleaned-up data with 12 topics.

- Print out the top terms for each topic.
- What is the coherence of the model with the c_v metric?

In [27]:
```
#Firstly Let's create the id2word Dictionary and corpus of words required for the LDA topic model

id2word = corpora.Dictionary(final_review_words)

corpus =[]
for review in final_review_words:
    new = id2word.doc2bow(review)
    corpus.append(new)

print(corpus[:20],"\n")
print("Number of reviews:",len(corpus),"\n")
print("Number of unique words:",len(id2word),"\n")

```

```
[[(0, 1), (1, 1)], [(2, 1), (3, 1), (4, 2), (5, 1), (6, 1), (7, 2), (8, 1), (9, 1), (10, 1), (11, 1)], [(12, 1)], [(2, 2), (13, 1), (14, 1)], [(15, 1), (16, 1)], [(17, 1), (18, 1)], [(4, 1), (19, 1)], [(14, 2), (20, 1), (21, 1), (22, 1), (23, 1)], [(24, 1), (25, 1), (26, 1), (27, 1)], [(28, 1)], [(5, 1), (29, 1), (30, 1)], [(4, 1), (31, 1), 2, 1)], [(4, 1), (14, 2), (30, 1), (32, 1), (33, 1)], [(3, 1), (34, 1), (35, 1), (36, 1)], [(37, 1)], [(38, 1)], [(2, 1), (39, 1), (40, 1), (41, 1), (42, 1), (43, 1), (44, 2), (45, 1), (46, 1)], [(4, 2), (32, 1), (47, 1)], [(48, 1)], [(14, 1), (49, 1), (50, 1)]]

Number of reviews: 11858

Number of unique words: 6341
```

In [28]:
```
#Let's Create a topic model using LDA on the cleaned-up data with 12 topics
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                            id2word=id2word,
                                            num_topics=12,
                                            random_state=42,
                                            update_every=1,
                                            chunksize=100,
                                            passes=10,
                                            alpha="auto")
```

In [29]:
```
lda_model.print_topics()
```

Out[29]:
```
[(0,
  '0.109*"money" + 0.072*"heat" + 0.062*"software" + 0.037*"value" + 0.034*"ram" + 0.034*"support" + 0.032*"game" + 0.026*"model" + 0.024*"contact" + 0.023*"headphone"'),
 (1,
  '0.269*"time" + 0.117*"charger" + 0.064*"turbo" + 0.037*"button" + 0.033*"effect" + 0.033*"power" + 0.030*"message" + 0.029*"number" + 0.027*"pic" + 0.024*"cam"'),
 (2,
  '0.159*"price" + 0.150*"performance" + 0.105*"network" + 0.064*"range" + 0.061*"speaker" + 0.034*"hr" + 0.029*"photo" + 0.023*"flash" + 0.021*"memory" + 0.016*"gb"'),
 (3,
  '0.411*"camera" + 0.163*"quality" + 0.032*"everything" + 0.024*"drain" + 0.023*"stock" + 0.021*"android" + 0.016*"image" + 0.014*"lag" + 0.014*"side" + 0.012*"earphone"'
 (4,
  '0.143*"processor" + 0.107*"use" + 0.107*"music" + 0.050*"core" + 0.031*"deca" + 0.030*"ko" + 0.027*"awesome" + 0.025*"gaming" + 0.022*"concern" + 0.021*"k5"'),
 (5,
  '0.194*"feature" + 0.111*"call" + 0.075*"sound" + 0.069*"option" + 0.034*"buy" + 0.030*"front" + 0.028*"thanks" + 0.022*"specification" + 0.021*"gallery" + 0.020*"voic
e"'),
 (6,
  '0.112*"month" + 0.096*"update" + 0.050*"bit" + 0.040*"replacement" + 0.040*"budget" + 0.040*"glass" + 0.034*"sensor" + 0.030*"company" + 0.028*"smartphone" + 0.028*"pur
ase"'),
 (7,
  '0.211*"battery" + 0.112*"problem" + 0.096*"issue" + 0.065*"day" + 0.042*"heating" + 0.037*"backup" + 0.027*"mode" + 0.026*"display" + 0.019*"video" + 0.017*"usage"'),
 (8,
  '0.105*"service" + 0.064*"delivery" + 0.060*"speed" + 0.052*"customer" + 0.051*"app" + 0.038*"system" + 0.032*"center" + 0.030*"refund" + 0.027*"hang" + 0.026*"need"'),
 (9,
  '0.152*"screen" + 0.094*"life" + 0.063*"charging" + 0.049*"card" + 0.042*"box" + 0.035*"cast" + 0.033*"expectation" + 0.025*"way" + 0.022*"today" + 0.019*"connectivity"'
 (10,
  '0.098*"device" + 0.089*"hour" + 0.064*"waste" + 0.063*"sim" + 0.044*"experience" + 0.041*"return" + 0.039*"data" + 0.033*"picture" + 0.033*"slot" + 0.032*"signal"'),
 (11,
  '0.162*"amazon" + 0.091*"charge" + 0.078*"lot" + 0.055*"mark" + 0.055*"work" + 0.052*"dolby" + 0.051*"apps" + 0.044*"min" + 0.026*"super" + 0.023*"class"')]
```

In [30]:
```
#Let's print the Top terms for each topic.

topics=[]
topic_terms=[]
for idx in range(12):
    topics.append("Topic "+ str(idx+1))
    terms=[]
    for term in lda_model.get_topic_terms(idx,topn=10):
        terms.append(id2word[term[0]])
    topic_terms.append(terms)

for idx in range(12):
    print(idx,topic_terms[idx])
```

```
0 ['money', 'heat', 'software', 'value', 'ram', 'support', 'game', 'model', 'contact', 'headphone']
1 ['time', 'charger', 'turbo', 'button', 'effect', 'power', 'message', 'number', 'pic', 'cam']
2 ['price', 'performance', 'network', 'range', 'speaker', 'hr', 'photo', 'flash', 'memory', 'gb']
3 ['camera', 'quality', 'everything', 'drain', 'stock', 'android', 'image', 'lag', 'side', 'earphone']
4 ['processor', 'use', 'music', 'core', 'deca', 'ko', 'awesome', 'gaming', 'concern', 'k5']
5 ['feature', 'call', 'sound', 'option', 'buy', 'front', 'thanks', 'specification', 'gallery', 'voice']
6 ['month', 'update', 'bit', 'replacement', 'budget', 'glass', 'sensor', 'company', 'smartphone', 'purchase']
7 ['battery', 'problem', 'issue', 'day', 'heating', 'backup', 'mode', 'display', 'video', 'usage']
8 ['service', 'delivery', 'speed', 'customer', 'app', 'system', 'center', 'refund', 'hang', 'need']
9 ['screen', 'life', 'charging', 'card', 'box', 'cast', 'expectation', 'way', 'today', 'connectivity']
10 ['device', 'hour', 'waste', 'sim', 'experience', 'return', 'data', 'picture', 'slot', 'signal']
11 ['amazon', 'charge', 'lot', 'mark', 'work', 'dolby', 'apps', 'min', 'super', 'class']
```

In [31]:
```
#Let's create a dataframe of the top terms in each topic
print('Topics DataFrame : ')
topics_df = pd.DataFrame(topic_terms).transpose()
topics_df.columns = topics
topics_df
```

```
Topics DataFrame :
```

Out[31]:

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10	Topic 11	Topic 12
0	money	time	price	camera	processor	feature	month	battery	service	screen	device	amazon
1	heat	charger	performance	quality	use	call	update	problem	delivery	life	hour	charge
2	software	turbo	network	everything	music	sound	bit	issue	speed	charging	waste	lot
3	value	button	range	drain	core	option	replacement	day	customer	card	sim	mark
4	ram	effect	speaker	stock	deca	buy	budget	heating	app	box	experience	work
5	support	power	hr	android	ko	front	glass	backup	system	cast	return	dolby
6	game	message	photo	image	awesome	thanks	sensor	mode	center	expectation	data	apps
7	model	number	flash	lag	gaming	specification	company	display	refund	way	picture	min
8	contact	pic	memory	side	concern	gallery	smartphone	video	hang	today	slot	super
9	headphone	cam	gb	earphone	k5	voice	purchase	usage	need	connectivity	signal	class

In [32]:
```
#Let's find the coherence of the model with the c_v metric

coherence_model_lda = CoherenceModel(model=lda_model, texts=final_review_words, dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('Coherence Score : ', coherence_lda)
```

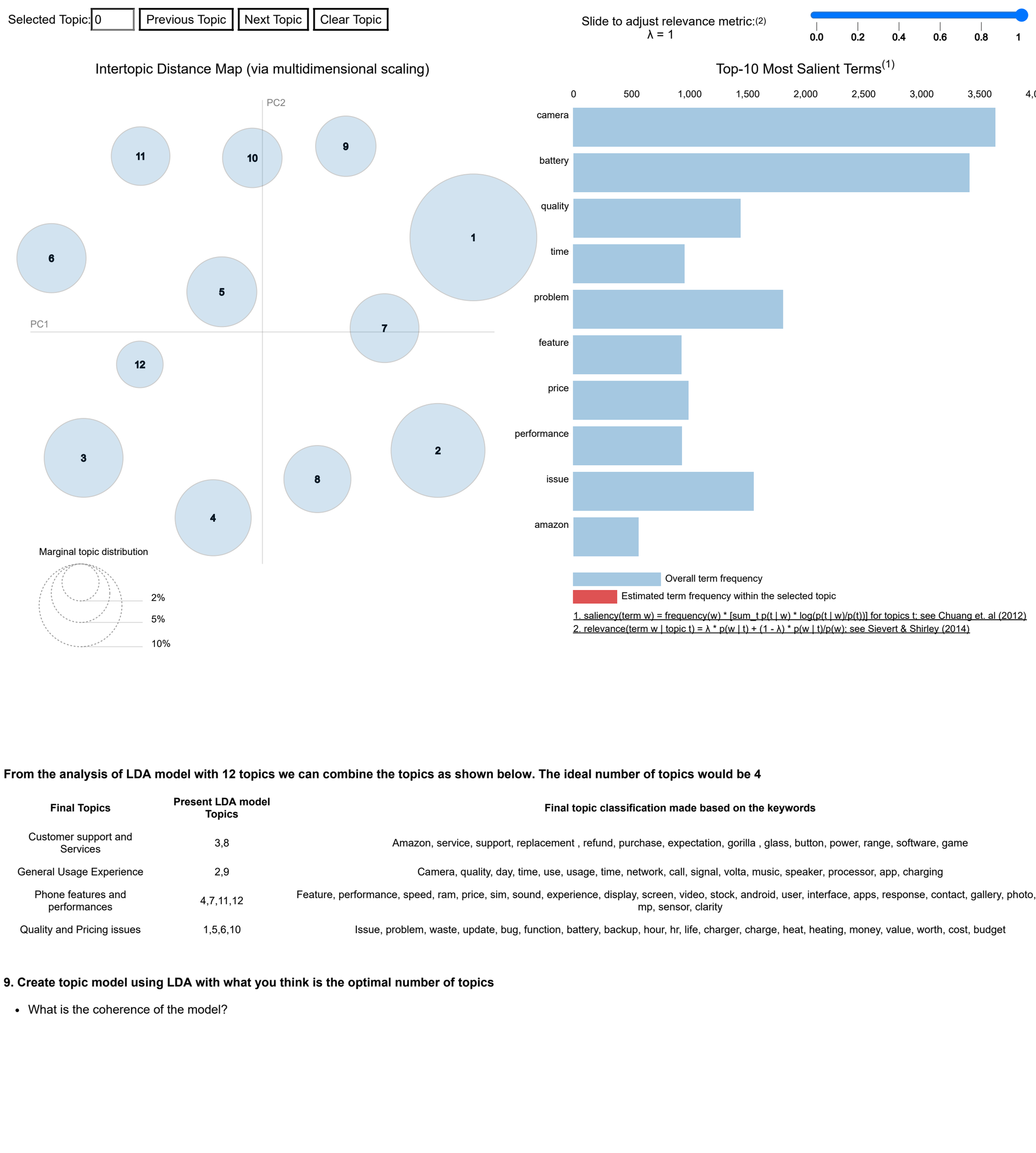
```
Coherence Score :  0.3818475081344903
```

9. Analyze the topics through the business lens.

- Determine which of the topics can be combined.

In [34]:
```
#Let's visualize the LDA Model using the Python library pyLDAvis for interactive topic model visualization
pyLDAvis.enable_notebook()
viz=pyLDAvis.gensim_models.prepare(lda_model,corpus,id2word,mds='mmds',R=10)
viz
```

Out[34]:



From the analysis of LDA model with 12 topics we can combine the topics as shown below. The ideal number of topics would be 4

Final Topics	Present LDA model Topics	Final topic classification made based on the keywords
Customer support and Services	3,8	Amazon, service, support, replacement , refund, purchase, expectation, gorilla , glass, button, power, range, software, game
General Usage Experience	2,9	Camera, quality, day, time, use, usage, time, network, call, signal, volta, music, speaker, processor, app, charging
Phone features and performances	4,7,11,12	Feature, performance, speed, ram, price, sim, sound, experience, display, screen, video, stock, android, user, interface, apps, response, contact, gallery, photo, flash, mp, sensor, clarity
Quality and Pricing issues	1,5,6,10	Issue, problem, waste, update, bug, function, battery, backup, hour, hr, life, charger, charge, heat, heating, money, value, worth, cost, budget

9. Create topic model using LDA with what you think is the optimal number of topics

- What is the coherence of the model?

In [35]:
```
#Let's Create a topic model using LDA  with 4 topics
lda_model_2 = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                              id2word=id2word,
                                              num_topics=4,
                                              random_state=47,
                                              update_every=1,
                                              chunksize=100,
                                              passes=10,
                                              alpha="auto")
```

In [36]:
```
lda_model_2.print_topics()
```

Out[36]:
```
[(0,
  '0.087*"problem" + 0.075*"issue" + 0.053*"time" + 0.034*"money" + 0.033*"heating" + 0.025*"update" + 0.022*"heat" + 0.019*"software" + 0.017*"charge" + 0.017*"waste"'),
 (1,
  '0.117*"camera" + 0.102*"battery" + 0.046*"quality" + 0.031*"day" + 0.031*"price" + 0.029*"performance" + 0.020*"network" + 0.018*"backup" + 0.016*"device" + 0.014*"hou
r"'),
 (2,
  '0.063*"feature" + 0.039*"amazon" + 0.037*"month" + 0.036*"call" + 0.032*"service" + 0.029*"charger" + 0.024*"sound" + 0.022*"option" + 0.020*"delivery" + 0.016*"bit"'),
 (3,
  '0.045*"screen" + 0.028*"life" + 0.019*"turbo" + 0.019*"charging" + 0.016*"ram" + 0.016*"work" + 0.016*"budget" + 0.016*"glass" + 0.014*"card" + 0.013*"sensor"')]
```

In [37]:
```
#Let's print the Top terms for each topic.

topics_model2=[]
topic_terms_model2=[]
for idx in range(4):
    topics_model2.append("Topic "+ str(idx+1))
    terms=[]
    for term in lda_model_2.get_topic_terms(idx,topn=10):
        terms.append(id2word[term[0]])
    topic_terms_model2.append(terms)

for idx in range(4):
    print(idx,topic_terms_model2[idx])
```

```
0 ['problem', 'issue', 'time', 'money', 'heating', 'update', 'heat', 'software', 'charge', 'waste']
1 ['camera', 'battery', 'quality', 'day', 'price', 'performance', 'network', 'backup', 'device', 'hour']
2 ['feature', 'amazon', 'month', 'call', 'service', 'charger', 'sound', 'option', 'delivery', 'bit']
3 ['screen', 'life', 'turbo', 'charging', 'ram', 'work', 'budget', 'glass', 'card', 'sensor']
```

In [38]:
```
#Let's create a dataframe of the top terms in each topic
topics_model_2_df= pd.DataFrame(topic_terms_model2).transpose()
topics_model_2_df.columns=topics_model2
topics_model_2_df
```

Out[38]:

	Topic 1	Topic 2	Topic 3	Topic 4
0	problem	camera	feature	screen
1	issue	battery	amazon	life
2	time	quality	month	turbo
3	money	day	call	charging
4	heating	price	service	ram
5	update	performance	charger	work
6	heat	network	sound	budget
7	software	backup	option	glass
8	charge	device	delivery	card
9	waste	hour	bit	sensor

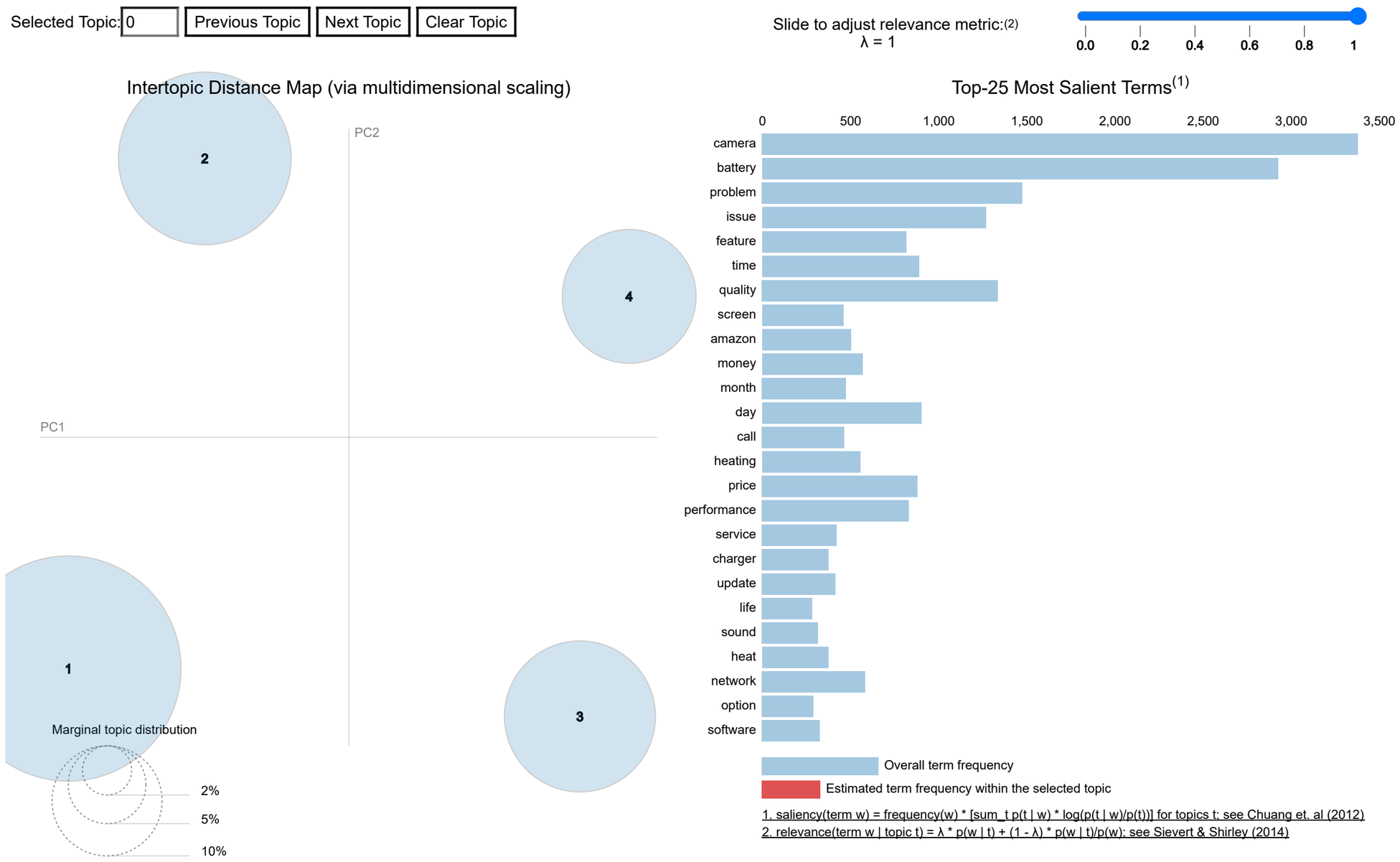
In [39]:
```
# Coherence of the new model

coherence_model_lda_2 = CoherenceModel(model=lda_model_2, texts=final_review_words, dictionary=id2word, coherence='c_v')
coherence_lda_2 = coherence_model_lda_2.get_coherence()
print('\nCoherence Score: ', coherence_lda_2)
```

```
Coherence Score:  0.510218984012
```

In [40]:
```
viz2 =pyLDAvis.gensim_models.prepare(lda_model_2,corpus,id2word,mds='mmds',R=25)
viz2
```

Out[40]:



11. The business should be able to interpret the topics.

- Name each of the identified topics.
- Create a table with the topic name and the top 10 terms in each to present to the business.

Name each of the identified topics.

Final Topics are as follows:

Topic1 = Problems and Issues (Based on the keywords : Problem, issue, heat, heating, time, waste, money, software, update, flash, charge, replacement, refund, customer, center, sim ,signal)

Topic2 = Usage Experience (Based on the keywords : Camera, quality, battery, backup, day, price, performance, speed, range, network, use, usage , speaker, music)

Topic3 = Customer Support and Service (Based on the keywords: Amazon, delivery, call, service, purchase, support, app, notification,system, photo, charger, sound, dolby, touch, memory, slot)

Topic4 = Hardware Specifications (Based on the keywords : Screen, life, turbo, charging, ram, budget, glass, gorilla, sensor, image, gallery, pic, color, headphone)

In [41]:
```
#let's Create a table with the final topic names and the top 10 terms in each to present to the  business.

Finaltopics= ["Problems and Issues","Usage Experience", "Customer support and Service", "Hardware Specifications"]
topics_model_2_df.columns=Finaltopics
topics_model_2_df
```

Out[41]:

	Problems and Issues	Usage Experience	Customer support and Service	Hardware Specifications
0	problem	camera	feature	screen
1	issue	battery	amazon	life
2	time	quality	month	turbo
3	money	day	call	charging
4	heating	price	service	ram
5	update	performance	charger	work
6	heat	network	sound	budget
7	software	backup	option	glass
8	charge	device	delivery	card
9	waste	hour	bit	sensor

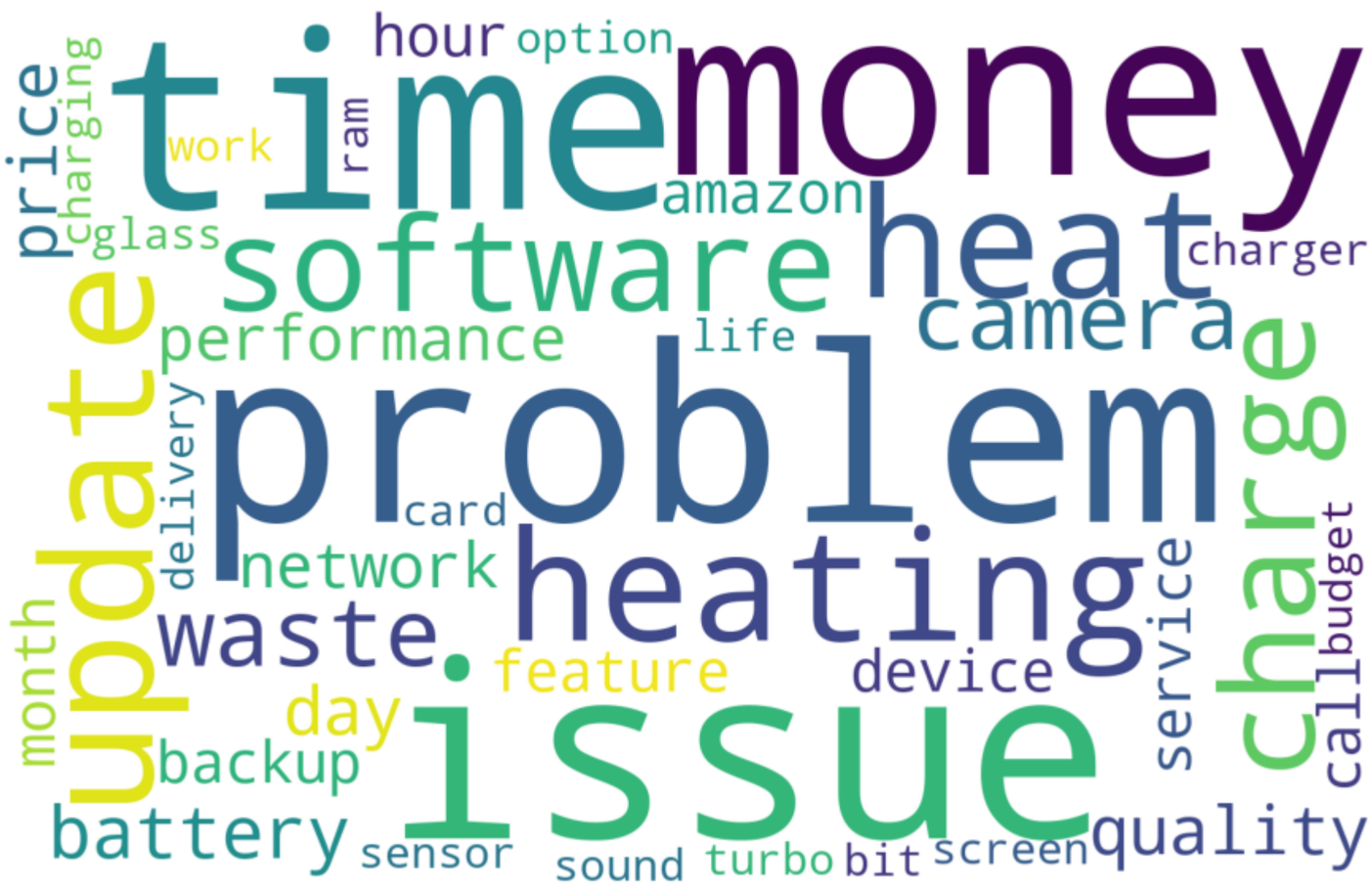
In [42]:
```
from wordcloud import WordCloud, STOPWORDS
topic_words = " "
stopwords = set(STOPWORDS)
tokens=[]
for topic in topic_terms_model2:
    for word in topic:
        tokens.append(word)

topic_words += " ".join(tokens)+" "

wordcloud = WordCloud(width = 1400, height = 900,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 8, max_font_size=250).generate(topic_words)

# plot the WordCloud image
plt.figure(figsize = (12, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```



---------- THANK YOU ----------

In [ ]:
```

```

In [ ]:
```

```