lenovo_topic_analysis_reviews

November 23, 2021

```
[ ]: import re
     import pandas as pd
     import nltk
     import gensim
     import matplotlib.pyplot as plt
     %matplotlib inline
     from wordcloud import WordCloud
```

```
[ ]: !pip install wordcloud
```

```
Requirement already satisfied: wordcloud in
/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/lib/python3.9/site-
packages (1.8.1)
Requirement already satisfied: numpy>=1.6.1 in
/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/lib/python3.9/site-
packages (from wordcloud) (1.21.4)
Requirement already satisfied: pillow in
/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/lib/python3.9/site-
packages (from wordcloud) (8.4.0)
Requirement already satisfied: matplotlib in
/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/lib/python3.9/site-
packages (from wordcloud) (3.5.0)
Requirement already satisfied: pyparsing>=2.2.1 in
/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/lib/python3.9/site-
packages (from matplotlib->wordcloud) (3.0.6)
Requirement already satisfied: python-dateutil>=2.7 in
/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/lib/python3.9/site-
packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: setuptools-scm>=4 in
/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/lib/python3.9/site-
packages (from matplotlib->wordcloud) (6.3.2)
Requirement already satisfied: fonttools>=4.22.0 in
/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/lib/python3.9/site-
packages (from matplotlib->wordcloud) (4.28.2)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/lib/python3.9/site-
packages (from matplotlib->wordcloud) (1.3.2)
Requirement already satisfied: packaging>=20.0 in
```

```
/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/lib/python3.9/site-
packages (from matplotlib->wordcloud) (21.3)
Requirement already satisfied: cycler>=0.10 in
/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/lib/python3.9/site-
packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: six>=1.5 in
/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/lib/python3.9/site-
packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Requirement already satisfied: tomli>=1.0.0 in
/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/lib/python3.9/site-
packages (from setuptools-scm>=4->matplotlib->wordcloud) (1.2.2)
Requirement already satisfied: setuptools in
/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/lib/python3.9/site-
packages (from setuptools-scm>=4->matplotlib->wordcloud) (58.1.0)
WARNING: You are using pip version 21.2.4; however, version 21.3.1 is
available.
You should consider upgrading via the
'/Users/sthuraisamy/code/watchblade.com/lab/ai-lab/.venv/bin/python -m pip
install --upgrade pip' command.
```

[ ]:
```
replace_vals = [(re.compile(r'@\w+'), ''),
                (re.compile(r'http\S+'), '')]
```

[ ]:
```
# common functions
```

[ ]:
```
def read_data_set(data_filename):
    '''function to read dataset and print some information about the dataset'''
    # read csv file into dataframe
    data_df = pd.read_csv(
        data_filename, delimiter=",", encoding="utf-8")

    # print the info of twitter data frame
    print(data_df.head())
    print(data_df.shape)
    print(data_df.columns)
    print(data_df.isnull().sum())

    return data_df
```

[ ]:
```
def pre_token_cleanup(text, replace_vals):
    '''function to pre-process the tweets'''
    text = text.lower()  # convert to lower case

    # text = replace_with(text, [('&amp;', 'and'), ('&gt;', '>'), ('&lt;',␣
 ↪'<')])
```

```
    for replace in replace_vals:
        text = re.sub(replace[0], replace[1], text)

    text = text.strip()  # remove leading and trailing whitespace

    return text
```

[ ]:
```
def clean_reviews(review_df):
    '''function to clean the reviews'''
    review_df["review"] = review_df["review"].apply(pre_token_cleanup,␣
↪args=(replace_vals,))
    print(review_df.head())


    return review_df
```

[ ]:
```
def view_wordcloud_common_words(review_df):
    '''function to view the common words'''
    # get the common words
    all_words = ','.join(list(review_df['review'].values))
    # print(all_words)

    # view the word cloud
    w_cloud = WordCloud(background_color='white', max_words=5000, width=600,␣
↪height=300, contour_width=3, contour_color='steelblue')
    w_cloud.generate(all_words)

    w_cloud.to_file('wordcloud.png')
```

[ ]:
```
def get_values_for_topic_analysis(review_df):
    review_values = review_df["review"].values

    return review_values
```

[ ]:
```
def get_word_tokens(review_values):
    '''function to get the word tokens'''
    word_tokens = []
    for review in review_values:
        word_tokens.append(nltk.word_tokenize(review))

    return word_tokens
```

[ ]:
```
# read the data from csv file into dataframe
review_df = read_data_set('../data/k8_review.csv')
```

```
   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…
(14675, 2)
Index(['sentiment', 'review'], dtype='object')
sentiment    0
review       0
dtype: int64
```

[ ]:
```
# Normalize casings and clean up the tweets
review_df = clean_reviews(review_df)

# extract the review text values into a list for easier manipulation.
review_values = get_values_for_topic_analysis(review_df)
print(review_values[:10])
```

```
   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…
['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 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 4000mah & booster
charger 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 me."
 '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 amazon is not returning the phone .highly disappointing of amazon'
 "only i'm telling don't buyi'm totally disappointedpoor batterypoor camerawaste
of money"
 'phone is awesome. but while charging, it heats up allot..really a genuine
reason to hate lenovo k8 note'
 'the battery level has worn down'
 "it's over hitting problems…and phone hanging problems lenovo k 8 note…so
where is service station in ahmedabad it's one years warranty so it's can change
the phone by lenovo"
 'a lot of glitches dont buy this thing better go for some other options']
```

[ ]:
```
view_wordcloud_common_words(review_df)
```

[ ]:
```
# tokenize the reviews using NLTK
review_word_tokens = get_word_tokens(review_values)
print(review_word_tokens[:10])
```

```
[['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', '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', '4000mah', '&', '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', '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', 'amazon', 'is', 'not', 'returning', 'the',
'phone', '.highly', 'disappointing', 'of', 'amazon'], ['only', 'i', "'m",
'telling', 'do', "n't", 'buyi', "'m", 'totally', 'disappointedpoor',
'batterypoor', 'camerawaste', 'of', 'money'], ['phone', 'is', 'awesome', '.',
'but', 'while', 'charging', ',', 'it', 'heats', 'up', 'allot', '..', 'really',
'a', 'genuine', 'reason', 'to', 'hate', 'lenovo', 'k8', 'note'], ['the',
'battery', 'level', 'has', 'worn', 'down'], ['it', "'s", 'over', 'hitting',
'problems', '…', 'and', 'phone', 'hanging', 'problems', 'lenovo', 'k', '8',
'note', '…', 'so', 'where', 'is', 'service', 'station', 'in', 'ahmedabad',
'it', "'s", 'one', 'years', 'warranty', 'so', 'it', "'s", 'can', 'change',
'the', 'phone', 'by', 'lenovo'], ['a', 'lot', 'of', 'glitches', 'dont', 'buy',
'this', 'thing', 'better', 'go', 'for', 'some', 'other', 'options']]
```

[ ]:
```python
# using NLTK to get the parts of speech of the sentences
review_sentences_postags = [nltk.pos_tag(sentence) for sentence in␣
 ↪review_word_tokens]
print(review_sentences_postags[:2])
```

```
[[('good', 'JJ'), ('but', 'CC'), ('need', 'VBP'), ('updates', 'NNS'), ('and',
'CC'), ('improvements', 'NNS')], [('worst', 'JJS'), ('mobile', 'NN'), ('i',
'NN'), ('have', 'VBP'), ('bought', 'VBN'), ('ever', 'RB'), (',', ','),
('battery', 'NN'), ('is', 'VBZ'), ('draining', 'VBG'), ('like', 'IN'), ('hell',
'NN'), (',', ','), ('backup', 'NN'), ('is', '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'), ('lie', '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'), ('4000mah', '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',
'CC'), ('go', 'VB'), ('for', 'IN'), ('this', 'DT'), ('else', 'JJ'), ('you',
'PRP'), ('will', 'MD'), ('regret', 'VB'), ('like', 'IN'), ('me', 'PRP'), ('.',
'.')]]
```

[ ]:
```
def get_postags_with_nouns(review_sentences_postags):
    '''function to get the pos tags with nouns'''
    noun_tags = ['NN', 'NNS', 'NNP', 'NNPS']
    postags_with_nouns = []
    for sentence in review_sentences_postags:
        postags_with_nouns.append([word for word, tag in sentence if tag in␣
 ↪noun_tags])

    return postags_with_nouns
```

[ ]:
```
# get the pos tags with nouns
postags_with_nouns = get_postags_with_nouns(review_sentences_postags)
print(postags_with_nouns[:10])
```

```
[['updates', 'improvements'], ['mobile', 'i', 'battery', 'hell', 'backup',
'hours', 'uses', 'idle', 'discharged.this', 'lie', 'amazon', 'lenove',
'battery', 'charger', 'hours', 'don'], ['i', '%', 'cash', '..'], [], ['phone',
'everthey', 'phone', 'problem', 'amazon', 'phone', 'amazon'], ['camerawaste',
'money'], ['phone', 'allot', '..', 'reason', 'k8'], ['battery', 'level'],
['problems', 'phone', 'hanging', 'problems', 'note', 'station', 'ahmedabad',
'years', 'phone', 'lenovo'], ['lot', 'glitches', 'thing', 'options']]
```

[ ]:
```
def get_postags_with_nouns_lemmed(postags_with_nouns):
    '''function to get the pos tags with nouns lemmatized'''
    lemmatizer = nltk.stem.WordNetLemmatizer()
    postags_with_nouns_lemmed = []
    for sentence in postags_with_nouns:
        postags_with_nouns_lemmed.append([lemmatizer.lemmatize(word) for word␣
 ↪in sentence])

    return postags_with_nouns_lemmed
```

[ ]:
```
# lemmatize the nouns
postags_with_nouns_lemmed = get_postags_with_nouns_lemmed(postags_with_nouns)
print(postags_with_nouns_lemmed[:10])
```

```
[['update', 'improvement'], ['mobile', 'i', 'battery', 'hell', 'backup', 'hour',
'us', 'idle', 'discharged.this', 'lie', 'amazon', 'lenove', 'battery',
'charger', 'hour', 'don'], ['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']]
```

```
[ ]: def remove_stop_words_and_puncs(postags_with_nouns_lemmed):
         '''function to remove stop words and punctuations'''
         stop_words = set(nltk.corpus.stopwords.words('english'))
         postags_with_nouns_lemmed_no_stop_words = []
         for sentence in postags_with_nouns_lemmed:
             postags_with_nouns_lemmed_no_stop_words.append([word for word in␣
      ↪sentence if word not in stop_words])

         return postags_with_nouns_lemmed_no_stop_words
```

```
[ ]: # Remove stopwords and punctuation (if there are any).
     postags_with_nouns_lemmed_no_stop_words =␣
      ↪remove_stop_words_and_puncs(postags_with_nouns_lemmed)
     print(postags_with_nouns_lemmed_no_stop_words[:10])
```

```
[['update', 'improvement'], ['mobile', 'battery', 'hell', 'backup', 'hour',
'us', 'idle', 'discharged.this', 'lie', 'amazon', 'lenove', 'battery',
'charger', 'hour'], ['%', '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']]
```

```
[ ]: def get_top_terms_for_topics_using_lda(postags_with_nouns_lemmed_no_stop_words,␣
      ↪num_topics, alpha, passes, workers):
         '''function to get the top terms for topics using LDA'''
         # Create a dictionary representation of the documents.
         dictionary = gensim.corpora.
      ↪Dictionary(postags_with_nouns_lemmed_no_stop_words)
         # Create a corpus from the bag of words.
         corpus = [dictionary.doc2bow(sentence) for sentence in␣
      ↪postags_with_nouns_lemmed_no_stop_words]
         # Build the LDA model.
         lda_model = gensim.models.LdaMulticore(corpus, num_topics=num_topics,␣
      ↪id2word=dictionary, passes=passes, alpha=alpha, random_state=426,␣
      ↪workers=workers)

         return lda_model, dictionary
```

```
[ ]: num_topics=12
     lda_model, dictionary =␣
      ↪get_top_terms_for_topics_using_lda(postags_with_nouns_lemmed_no_stop_words,␣
      ↪num_topics=num_topics, passes=20,alpha='symmetric', workers=3)
```

```
print(lda_model.print_topics(num_topics=num_topics, num_words=10))
```

```
[(0, '0.058*"call" + 0.054*"screen" + 0.041*"sim" + 0.039*"glass" +
0.019*"gorilla" + 0.019*"jio" + 0.018*"voice" + 0.017*"time" + 0.016*"display" +
0.016*"volta"'), (1, '0.306*"battery" + 0.080*"backup" + 0.043*"hour" +
0.035*"time" + 0.033*"life" + 0.029*"drain" + 0.027*"day" + 0.025*"problem" +
0.020*"charge" + 0.018*"heat"'), (2, '0.201*"camera" + 0.185*"product" +
0.106*"quality" + 0.023*"performance" + 0.020*"phone" + 0.018*"waste" +
0.014*"money" + 0.013*"clarity" + 0.012*"picture" + 0.012*"mark"'), (3,
'0.043*"hai" + 0.038*"h" + 0.021*"box" + 0.017*"ho" + 0.016*"item" +
0.012*"cable" + 0.011*"review" + 0.011*"lenovo" + 0.010*"ka" + 0.010*"model"'),
(4, '0.254*"phone" + 0.035*"lenovo" + 0.020*"time" + 0.018*"note" +
0.016*"issue" + 0.014*"update" + 0.013*"feature" + 0.011*"software" +
0.009*"amazon" + 0.009*"month"'), (5, '0.036*"phone" + 0.035*"amazon" +
0.022*"return" + 0.020*"smartphone" + 0.020*"option" + 0.017*"product" +
0.016*"screen" + 0.014*"ram" + 0.013*"app" + 0.013*"processor"'), (6,
'0.097*"phone" + 0.054*"camera" + 0.038*"speaker" + 0.035*"battery" +
0.020*"everything" + 0.019*"sound" + 0.017*"budget" + 0.015*"quality" +
0.013*"headphone" + 0.013*"thing"'), (7, '0.312*"mobile" + 0.039*"…" +
0.035*"superb" + 0.034*"*" + 0.025*"delivery" + 0.021*"worth" + 0.014*"awesome"
+ 0.014*"money" + 0.014*"…" + 0.014*"feature"'), (8, '0.192*"problem" +
0.125*"issue" + 0.087*"heating" + 0.080*"network" + 0.023*"handset" +
0.013*"month" + 0.013*"connectivity" + 0.013*"ok" + 0.010*"buy" +
0.010*"signal"'), (9, '0.091*"note" + 0.057*"k8" + 0.045*"service" +
0.023*"phone" + 0.021*"lenovo" + 0.018*"day" + 0.015*"customer" +
0.015*"product" + 0.014*"amazon" + 0.013*"battery"'), (10, '0.257*".." +
0.106*"…" + 0.063*"money" + 0.062*"phone" + 0.046*"performance" +
0.039*"value" + 0.012*"…" + 0.009*"plz" + 0.008*"amazon" + 0.007*"camera"'),
(11, '0.100*"phone" + 0.063*"price" + 0.047*"camera" + 0.028*"charger" +
0.025*"device" + 0.024*"issue" + 0.023*"feature" + 0.023*"range" + 0.020*"day" +
0.016*"battery"')]
```

```
[ ]: def get_coherence_score_using_lda(lda_model, dictionary,␣
      ↪postags_with_nouns_lemmed_no_stop_words):
         '''function to get the coherence score using LDA'''

         # Compute Coherence Score
         coherence_model_lda = gensim.models.CoherenceModel(model=lda_model,␣
      ↪texts=postags_with_nouns_lemmed_no_stop_words, dictionary=dictionary,␣
      ↪coherence='c_v')
         coherence_lda = coherence_model_lda.get_coherence()

         return coherence_lda
```

```
[ ]: coherence_lda = get_coherence_score_using_lda(lda_model, dictionary,␣
      ↪postags_with_nouns_lemmed_no_stop_words)
     print('Coherence score: ',coherence_lda)
```

```
Coherence score:  0.5323107049732453
```

```
[ ]: def␣
      ↪get_coherence_score_for_multiple_topics(postags_with_nouns_lemmed_no_stop_words,␣
      ↪num_topics_list):
         '''function to get the coherence score for multiple topics'''
         coherence_scores = []
         for num_topics in num_topics_list:

             lda_model, dictionary =␣
      ↪get_top_terms_for_topics_using_lda(postags_with_nouns_lemmed_no_stop_words,␣
      ↪num_topics=num_topics, passes=30,alpha='symmetric', workers=5)
             coherence_model_lda = get_coherence_score_using_lda(lda_model,␣
      ↪dictionary, postags_with_nouns_lemmed_no_stop_words, num_topics)
             coherence_scores.append(coherence_model_lda)

         return coherence_scores
```

```
[ ]: num_topics_list = [5,6,7,8,9,10]
     coherence_scores =␣
      ↪get_coherence_score_for_multiple_topics(postags_with_nouns_lemmed_no_stop_words,␣
      ↪num_topics_list)
     print(coherence_scores)
```

```
[0.5189486456095401, 0.5142553731846591, 0.5045417075996442, 0.5687858691150023,
0.5407228460090576, 0.5400228379980615]
```

```
[ ]: # get the model for better coherence score
     num_topics_for_better_coherence = num_topics_list[coherence_scores.
      ↪index(max(coherence_scores))]
     print('Number of topics for better coherence score:␣
      ↪',num_topics_for_better_coherence)
     lda_model_v1, dictionary_v1 =␣
      ↪get_top_terms_for_topics_using_lda(postags_with_nouns_lemmed_no_stop_words,␣
      ↪num_topics=num_topics_for_better_coherence, passes=30,alpha='symmetric',␣
      ↪workers=3)

     better_coherence_model_lda = get_coherence_score_using_lda(lda_model_v1,␣
      ↪dictionary_v1, postags_with_nouns_lemmed_no_stop_words,␣
      ↪num_topics_for_better_coherence)
     print('Better coherence model: ',better_coherence_model_lda)
```

```
Number of topics for better coherence score:  8
Better coherence model:  0.5521932218121997
```

```
[ ]: def print_topics_report(final_lda_model):
         topic_words = {}
         for idx, topic in final_lda_model.print_topics(-1):
```

```
        temp = []
        for item in topic.split('+'):
            item_alpha = [letter for letter in item if letter.isalpha()]
            temp.append(''.join(item_alpha))
        topic_words[('Topic_'+str(idx+1))] = temp

    topics_df = pd.DataFrame(topic_words)
    topics_df.index = ['Word_'+str(i+1) for i in range(topics_df.shape[0])]
    print(topics_df)
```

[ ]:
```
print_topics_report(lda_model_v1)
```

```
         Topic_1     Topic_2      Topic_3  Topic_4  Topic_5    Topic_6  \
Word_1      call     problem       camera  charger    phone      price
Word_2    screen      device      product      hai     note      phone
Word_3     glass       super      quality        h   lenovo      range
Word_4       sim   excellent        phone    turbo    issue    feature
Word_5    option          ok  performance      box     time     camera
Word_6      time         set                     k               music
Word_7   network       dolby         mode     note  product        ram
Word_8       jio     product        depth   charge  service  processor
Word_9      cast       atmos                    ho        k       note
Word_10  gorilla  connection      feature     item      day     memory

             Topic_7   Topic_8
Word_1       battery    mobile
Word_2         phone
Word_3        camera     money
Word_4         issue
Word_5       problem   product
Word_6        backup     waste
Word_7       heating     value
Word_8   performance
Word_9          life  delivery
Word_10          day    superb
```

[ ]:
```
```