**Aspect Based Sentiment Analysis in Spark NLP**[¶](#gjdgxs)

#### Model Details: <https://nlp.johnsnowlabs.com/2020/12/29/ner_aspect_based_sentiment_en.html>[¶](#30j0zll)

Spark NLP documentation and instructions:[¶](#1fob9te)

<https://nlp.johnsnowlabs.com/docs/en/quickstart>

### You can find details about Spark NLP annotators here:[¶](#3znysh7)

<https://nlp.johnsnowlabs.com/docs/en/annotators>

### You can find details about Spark NLP models here:[¶](#2et92p0)

<https://nlp.johnsnowlabs.com/models>

1. Colab Setup[¶](#tyjcwt)

Install Dependencies and Libraries

In [ ]:

*# Install PySpark and Spark NLP*  
*#! pip install -q pyspark==3.1.2 spark-nlp*  
  
*# Install Spark NLP Display lib*  
*#! pip install --upgrade -q spark-nlp-display*

In [ ]:

In [ ]:

*#import gc*

In [ ]:

*#gc.collect()*

ERROR:py4j.java\_gateway:An error occurred while trying to connect to the Java server (127.0.0.1:35511)  
Traceback (most recent call last):  
 File "/usr/local/lib/python3.7/dist-packages/py4j/java\_gateway.py", line 977, in \_get\_connection  
 connection = self.deque.pop()  
IndexError: pop from an empty deque  
  
During handling of the above exception, another exception occurred:  
  
Traceback (most recent call last):  
 File "/usr/local/lib/python3.7/dist-packages/py4j/java\_gateway.py", line 1115, in start  
 self.socket.connect((self.address, self.port))  
ConnectionRefusedError: [Errno 111] Connection refused

Out[ ]:

1413

Import and start the Spark session

In [ ]:

**import** pandas **as** pd  
**from** pyspark.ml **import** Pipeline  
**from** pyspark.sql **import** SparkSession  
**import** pyspark.sql.functions **as** F  
  
**import** sparknlp  
**from** sparknlp.annotator **import** **\***  
**from** sparknlp.base **import** **\***  
  
spark **=** sparknlp**.**start()  
  
*# manually start session*  
'''  
spark = SparkSession.builder \  
 .appName('Spark NLP Licensed') \  
 .master('local[\*]') \  
 .config('spark.driver.memory', '16G') \  
 .config('spark.serializer', 'org.apache.spark.serializer.KryoSerializer') \  
 .config('spark.kryoserializer.buffer.max', '2000M') \  
 .config('spark.jars.packages', 'com.johnsnowlabs.nlp:spark-nlp\_2.11:' +sparknlp.version()).getOrCreate()  
'''

Out[ ]:

"\nspark = SparkSession.builder .appName('Spark NLP Licensed') .master('local[\*]') .config('spark.driver.memory', '16G') .config('spark.serializer', 'org.apache.spark.serializer.KryoSerializer') .config('spark.kryoserializer.buffer.max', '2000M') .config('spark.jars.packages', 'com.johnsnowlabs.nlp:spark-nlp\_2.11:' +sparknlp.version()).getOrCreate()\n"

2. Build Pipeline[¶](#3dy6vkm)

In [ ]:

document\_assembler **=** DocumentAssembler() \  
 **.**setInputCol('text')\  
 **.**setOutputCol('document')  
  
sentence\_detector **=** SentenceDetector() \  
 **.**setInputCols(['document'])\  
 **.**setOutputCol('sentence')  
  
tokenizer **=** Tokenizer()\  
 **.**setInputCols(['sentence']) \  
 **.**setOutputCol('token')  
  
word\_embeddings **=** WordEmbeddingsModel**.**pretrained("glove\_6B\_300", "xx")\  
 **.**setInputCols(["document", "token"])\  
 **.**setOutputCol("embeddings")  
   
ner\_model **=** NerDLModel**.**pretrained("ner\_aspect\_based\_sentiment")\  
 **.**setInputCols(["document", "token", "embeddings"])\  
 **.**setOutputCol("ner")  
  
ner\_converter **=** NerConverter()\  
 **.**setInputCols(['sentence', 'token', 'ner']) \  
 **.**setOutputCol('ner\_chunk')  
  
nlp\_pipeline **=** Pipeline(stages**=**[  
 document\_assembler,   
 sentence\_detector,  
 tokenizer,  
 word\_embeddings,  
 ner\_model,  
 ner\_converter])  
  
empty\_df **=** spark**.**createDataFrame([['']])**.**toDF('text')  
pipeline\_model **=** nlp\_pipeline**.**fit(empty\_df)  
light\_pipeline **=** LightPipeline(pipeline\_model)

glove\_6B\_300 download started this may take some time.  
Approximate size to download 426.2 MB  
[OK!]  
ner\_aspect\_based\_sentiment download started this may take some time.  
Approximate size to download 21.3 MB  
[OK!]

In [ ]:

'''  
from google.colab import auth  
auth.authenticate\_user()  
  
# https://cloud.google.com/resource-manager/docs/creating-managing-projects  
project\_id = 'iconic-being-343500'  
!gcloud config set project {project\_id}  
'''

Updated property [core/project].

In [ ]:

'''  
# Download the file from a given Google Cloud Storage bucket.  
bucket\_name='datasetsbdp'  
!gsutil cp gs://{bucket\_name}/dataset\_review.json /content/sample\_data/dataset\_review.json  
   
# Print the result to make sure the transfer worked.  
#!cat /tmp/gsutil\_download.json  
'''

Copying gs://datasetsbdp/dataset\_review.json...  
\ [1 files][ 5.0 GiB/ 5.0 GiB] 48.7 MiB/s   
Operation completed over 1 objects/5.0 GiB.

In [ ]:

In [ ]:

**import** pandas **as** pd

In [ ]:

file\_name **=** "/content/genotext\_test"

In [ ]:

review**=**pd**.**read\_csv(file\_name,header**=None**,index\_col**=False**)

In [ ]:

review**.**columns **=**['text']

In [ ]:

review

Out[ ]:

|  | **text** |
| --- | --- |
| **0** | Meh Not too impressed So I had to do the Pat ... |
| **1** | We arrived here just after midnight and had no... |
| **2** | Both the cheese steak and the freedom fries we... |
| **3** | Kind of disappointed that after waiting min... |
| **4** | Yeah there s a rivalry with Pat s Yeah they... |
| **...** | ... |
| **3423** | So I believe in schemes and conspiracy theorie... |
| **3424** | Let s see what adjectives come to mind when I ... |
| **3425** | We love this place My family comes here ever... |
| **3426** | First of All for a crappy cheese steak ... |
| **3427** | I was told to avoid Genos and Pats as they are... |

3428 rows × 1 columns

3. Create example inputs[¶](#1t3h5sf)

In [ ]:

*#input\_list=[text\_list[2]]*

In [ ]:

*#input\_list*

Out[ ]:

['Both the cheese steak and the freedom fries were not that tasty I was a bit disappointed']

In [ ]:

'''  
# Enter examples as strings in this array  
input\_list = [  
 """From the beginning, we were met by friendly staff members, and the convienent parking at Chelsea Piers made it easy for us to get to the boat."""]  
'''

4. Run the pipeline[¶](#4d34og8)

Full Pipeline (Expects a spark Data Frame)

In [ ]:

*#text\_list[0]*

In [ ]:

*#input\_list*

Out[ ]:

['Both the cheese steak and the freedom fries were not that tasty I was a bit disappointed']

In [ ]:

*#df = spark.createDataFrame(pd.DataFrame({"text": input\_list}))*

In [ ]:

df **=** spark**.**createDataFrame(pd**.**DataFrame(review))

In [ ]:

result **=** pipeline\_model**.**transform(df)

In [ ]:

result**.**show(5)

+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+  
| text| document| sentence| token| embeddings| ner| ner\_chunk|  
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+  
|Meh Not too impr...|[{document, 0, 12...|[{document, 0, 12...|[{token, 0, 2, Me...|[{word\_embeddings...|[{named\_entity, 0...|[{chunk, 42, 44, ...|  
|We arrived here j...|[{document, 0, 61...|[{document, 0, 61...|[{token, 0, 1, We...|[{word\_embeddings...|[{named\_entity, 0...|[{chunk, 68, 72, ...|  
|Both the cheese s...|[{document, 0, 88...|[{document, 0, 88...|[{token, 0, 3, Bo...|[{word\_embeddings...|[{named\_entity, 0...|[{chunk, 9, 20, c...|  
|Kind of disappoin...|[{document, 0, 84...|[{document, 0, 84...|[{token, 0, 3, Ki...|[{word\_embeddings...|[{named\_entity, 0...|[{chunk, 32, 38, ...|  
|Yeah there s a r...|[{document, 0, 38...|[{document, 0, 38...|[{token, 0, 3, Ye...|[{word\_embeddings...|[{named\_entity, 0...| []|  
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+  
only showing top 5 rows

Light Pipeline (Expects a list of string)

In [ ]:

'''  
lresult = light\_pipeline.fullAnnotate(input\_list)  
'''

5. Visualize results[¶](#2s8eyo1)

Full Pipeline Result

In [ ]:

'''  
# Using display lib  
from sparknlp\_display import NerVisualizer  
  
NerVisualizer().display(result.collect()[0], 'ner\_chunk', 'document')  
'''

Both the cheese steak NEG and the freedom fries NEG were not that tasty I was a bit disappointed

In [ ]:

*# Process manually*  
exploded **=** F**.**explode(F**.**arrays\_zip('ner\_chunk.result', 'ner\_chunk.metadata'))  
select\_expression\_0 **=** F**.**expr("cols['0']")**.**alias("chunk")  
select\_expression\_1 **=** F**.**expr("cols['1']['entity']")**.**alias("ner\_label")  
aspect**=**result**.**select(exploded**.**alias("cols")) \  
 **.**select(select\_expression\_0, select\_expression\_1)  
  
*#result = result.toPandas()*

+---------------+---------+  
| chunk|ner\_label|  
+---------------+---------+  
| Pat| NEG|  
| onions| NEG|  
| bread| NEG|  
| meat| NEG|  
| meat| NEG|  
| bread| NEG|  
| Pat| NEG|  
| place| NEG|  
| tables| NEG|  
|chucks of steak| POS|  
| cheese| NEG|  
| flavor| NEG|  
| prices| NEG|  
| flavor| NEG|  
| cheese steak| NEG|  
| fries| NEG|  
| waiting| NEG|  
| food| POS|  
| mushroom| NEG|  
| bread| NEG|  
+---------------+---------+  
only showing top 20 rows

In [ ]:

aspect**.**select("chunk")**.**groupby("chunk")**.**count()**.**show()

+--------------------+-----+  
| chunk|count|  
+--------------------+-----+  
| orange color| 1|  
| amoroso rolls| 1|  
| peppers Bread| 1|  
| Dr Pepper| 1|  
| crust| 3|  
| staff| 135|  
|mushrooms freshly...| 1|  
| Jims| 3|  
| ribbon of ketchup| 1|  
| rolls| 21|  
| brusque| 3|  
|bun with sliced s...| 1|  
| chesesteak| 1|  
| Beef sandwich| 1|  
| television| 2|  
| Philly CheeseSteak| 1|  
| meat portion| 1|  
| richness| 1|  
| chees wiz| 1|  
| Pats steak| 1|  
+--------------------+-----+  
only showing top 20 rows

Light Pipeline Result

In [ ]:

aspect\_count**=**aspect**.**groupBy('chunk','ner\_label')**.**count()**.**sort('count')**.**orderBy(['count'],ascending**=**[0])

In [ ]:

aspect\_count**.**show(20)

+------------+---------+-----+  
| chunk|ner\_label|count|  
+------------+---------+-----+  
| meat| NEG| 520|  
| bread| NEG| 437|  
| steak| NEG| 433|  
| Pat| NEG| 391|  
| Geno s| POS| 294|  
| Geno| NEG| 275|  
| food| NEG| 264|  
|cheesesteaks| POS| 227|  
| fries| NEG| 226|  
| sandwich| NEG| 220|  
| steak| POS| 184|  
| onions| POS| 179|  
| cheese| NEG| 166|  
| service| NEG| 163|  
| bread| POS| 154|  
|cheese steak| POS| 154|  
| onions| NEG| 146|  
| cheesesteak| POS| 145|  
| Geno s| NEG| 144|  
| place| NEG| 139|  
+------------+---------+-----+  
only showing top 20 rows

In [ ]:

aspect\_count**.**filter(aspect\_count**.**ner\_label**==**'NEG')**.**show(20)

+------------+---------+-----+  
| chunk|ner\_label|count|  
+------------+---------+-----+  
| meat| NEG| 520|  
| bread| NEG| 437|  
| steak| NEG| 433|  
| Pat| NEG| 391|  
| Geno| NEG| 275|  
| food| NEG| 264|  
| fries| NEG| 226|  
| sandwich| NEG| 220|  
| cheese| NEG| 166|  
| service| NEG| 163|  
| onions| NEG| 146|  
| Geno s| NEG| 144|  
| place| NEG| 139|  
| staff| NEG| 99|  
|cheese steak| NEG| 96|  
| flavor| NEG| 88|  
| steaks| NEG| 85|  
| cheesesteak| NEG| 85|  
| sandwiches| NEG| 77|  
| Pat s| NEG| 65|  
+------------+---------+-----+  
only showing top 20 rows

In [ ]:

aspect\_count**.**filter(aspect\_count**.**ner\_label**==**'POS')**.**show(20)

+-------------+---------+-----+  
| chunk|ner\_label|count|  
+-------------+---------+-----+  
| Geno s| POS| 294|  
| cheesesteaks| POS| 227|  
| steak| POS| 184|  
| onions| POS| 179|  
| bread| POS| 154|  
| cheese steak| POS| 154|  
| cheesesteak| POS| 145|  
| fries| POS| 130|  
| meat| POS| 123|  
| Geno| POS| 117|  
| cheese| POS| 105|  
| sandwich| POS| 104|  
| service| POS| 97|  
| food| POS| 95|  
| steaks| POS| 77|  
| place| POS| 61|  
| cheese fries| POS| 52|  
|Geno s Steaks| POS| 44|  
| Pat| POS| 44|  
|cheese steaks| POS| 44|  
+-------------+---------+-----+  
only showing top 20 rows

In [ ]:

*# Using display lib*  
**from** sparknlp\_display **import** NerVisualizer  
  
NerVisualizer()**.**display(lresult[0], 'ner\_chunk', 'document')

Both the cheese steak NEG and the freedom fries NEG were not that tasty I was a bit disappointed

In [ ]:

'''  
# Process manually  
for example in lresult:  
 for res in example['ner\_chunk']:  
 print ('Token/Phrase:', res.result, 'Sentiment: ', res.metadata['entity'])  
'''

Token/Phrase: cheese steak Sentiment: NEG  
Token/Phrase: fries Sentiment: NEG