SparkNLP

TECH REVIEW

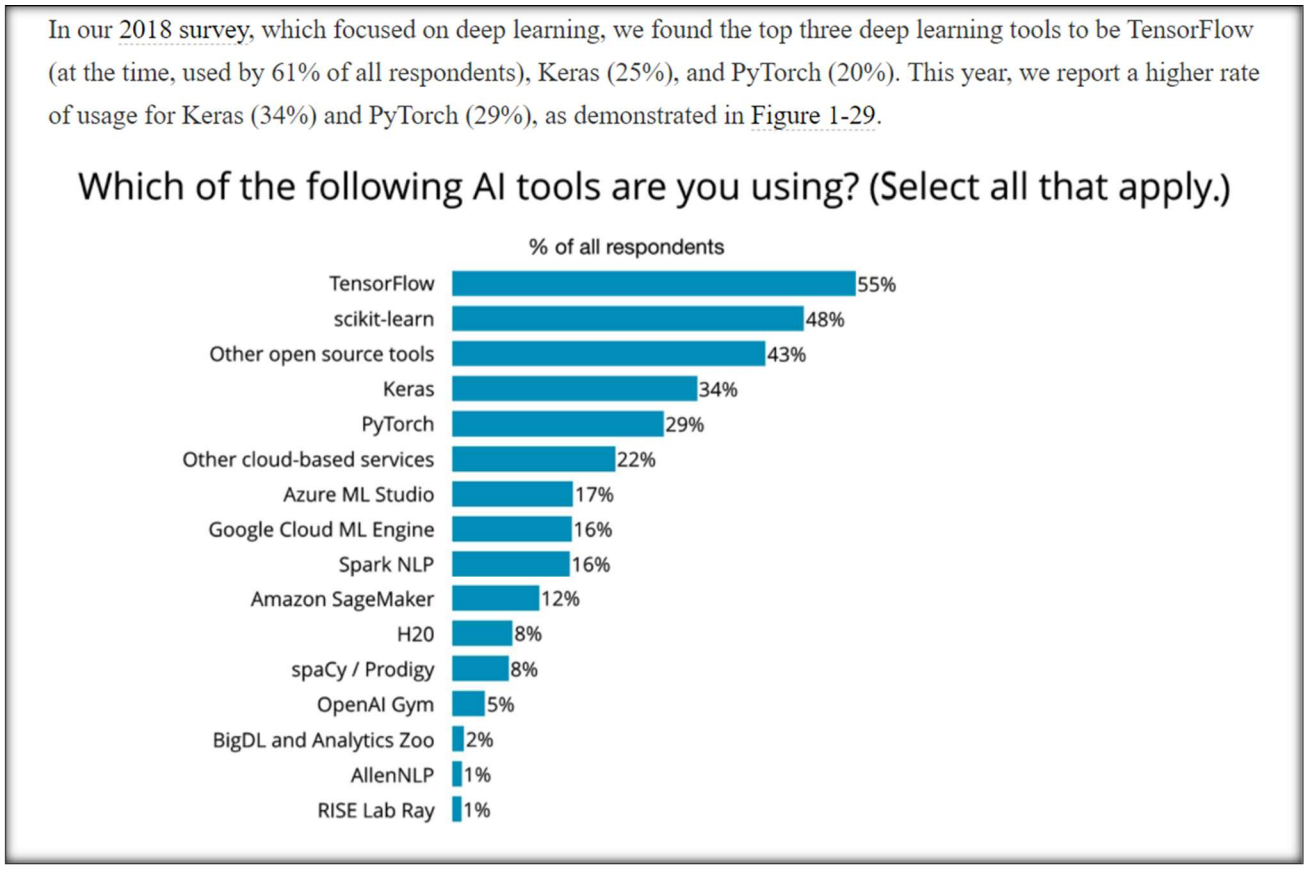
Diptam Sarkar | Text Information System | Nov 15, 2020

What is Spark NLP

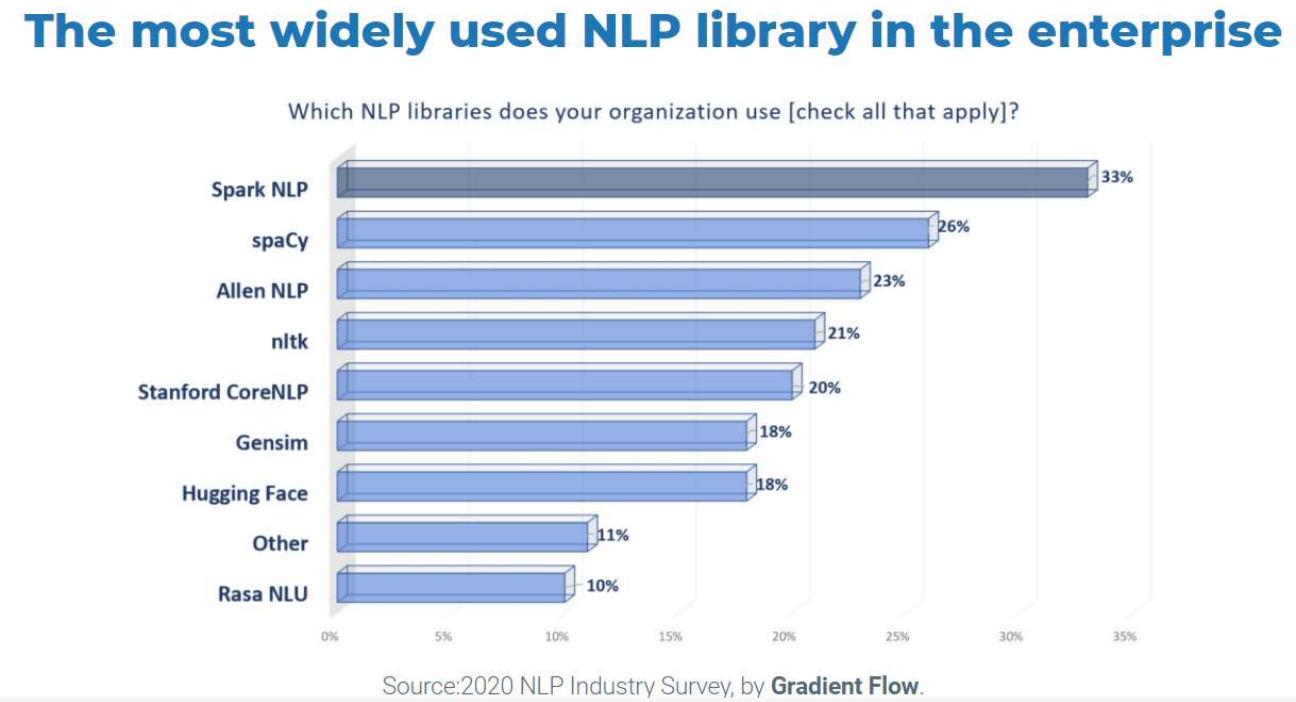
Spark NLP is an open source state of the art natural language processing library by Jon Snow labs that recently getting lots of buzz in tech industry. Many enterprises like Capital one, Docu-sign, Verizon is investing a good amount of resources. Along with that three of the major cloud computing vendor Amazon, Google and Microsoft both working on offering a full-scale spark nlp based service through their platform.

In a recent survey by OReilly it showed that enterprises are in growing trend of adopting AI and spark nlp listed as 7th most popular library and mighty good where we have all the heavy weights of the industry like Keras, Tensorflow and PyTorch.

In our 2018 survey, which focused on deep learning, we found the top three deep learning tools to be TensorFlow (at the time, used by 61% of all respondents), Keras (25%), and PyTorch (20%). This year, we report a higher rate of usage for Keras (34%) and PyTorch (29%), as demonstrated in Figure 1-29.



Also, its right now most standalone open source NLP library topping spacy or even nltk



Idea behind it

Big selling point of spark nlp is obviously that it is backed by Apache Spark, Apache Spark is general purpose in memory distributed data processing engine. Its immensely popular & gained a lot of focus from enterprises in recent years. Spark is known for two things, one its very fast and second its very easily scalable. Another big thing is its supports both Java/Scala and Python natively. Spark already had its own machine learning library SparkML and few other modules which can be used for certain NLP tasks but not many.

So as an enterprise if they are trying to implement a full-blown solution in their architectural flow a scalable nlp mechanism with Apache Spark, normally they had to mix and match with other solutions, but its normally considered inefficient if you are using different tech stack between data processing and nlp. Spark nlp is trying cover that gap between spark and sparkML where you are essentially getting full structured nlp library that work seamlessly with Apache Spark.

Who built it?

John Snow Labs is an award-winning data analytics and AI research company who works in healthcare industry. They developed sparkNLP, The company provides commercial support, indemnification and consulting for it. This provides the library with long-term financial backing, a funded active development team, and a growing stream of real-world projects that drives robustness and roadmap prioritization.

What does it offer?

Spark nlp is built with Spark and Tensorflow in back, its written in scala. Its offers a single unified solution for all our NLP needs which is by their words highly scalable and highly accurate. It has multi language support, reading their whole documentation I found out thar it has multi language supports which supports most of the work major languages, like English, French, German, Greek even many Indian languages like Marathi, Bengali. (I will keep separate section below language support as many of feature is experimental for many languages)

This what we get out of the box,

- Tokenization
- Stop Words Removal
- Normalizer
- Stemmer
- Lemmatizer
- NGrams
- Regex Matching
- Text Matching
- Chunking
- Date Matcher
- **Part-of-speech** tagging
- Sentence Detector (DL models)
- Dependency parsing (Labeled/unlabled)
- **Sentiment** Detection (ML models)
- **Spell** Checker (ML and DL models)
- Word Embeddings (**GloVe** and **Word2Vec**)
- **BERT** Embeddings
- **ELMO** Embeddings
- **Universal Sentence** Encoder
- **BERT Sentence** Embeddings
- **Sentence** Embeddings
- **Chunk** Embeddings
- Unsupervised **keywords extraction**
- Language **Detection** & **Identification**
- Multi-class Text **Classification** (DL model)
- Multi-label Text **Classification** (DL model)
- Multi-class **Sentiment Analysis** (DL model)
- **Named entity** recognition (DL model)
- Easy **TensorFlow** integration
- Full integration with **Spark ML** functions
- **330+** pre-trained **models** in **46 languages!**
- **90+** pre-trained **pipelines** in **13 languages!**

Another big feature of spark nlp is that it offers almost 400 trained model which comes with open source version. They are plug and play model, means you can use any of model you want to in your setup and they will work right out of the box. I will explore couple of them via by local setup.

It also has two paid service, one being spark nlp for healthcare and another being spark OCR.

Spark NLP for Healthcare is a commercial extension of Spark NLP for clinical and biomedical text mining.

Spark OCR is commercial extension of Spark NLP for optical character recognition from images, scanned PDF documents, and DICOM files

What & Why I am reviewing it?

I am a nlp newbie, text information course is my first encounter with perspective of language processing. We use Meta is our in our course, it is showing its age. Its due for an upgrade. While is use of Meta is sufficient what our course offers, two things are unavoidable one being its lack of proper documentation, two is programming language support. I am veteran Java developer, although I am quite comfortable with Python myself, I have seen many java people struggle with Python initially coming from such statically typed strong verbose language like Java. So, having flexibility of use of Java or Scala can be fantastic for many people.

Upon my research I found there are not many libraries out there that offers this flexibility of choice between Java or Python. But spark nlp does as it was written with Scala. So, I am trying to find out through my review if Spark nlp can be a no cost alternative to Meta which can also give people a choice of language.

Another thing was quite exiting for me, was having Bengali language support. I am natively Bengali speaker; it is my mother tongue. There are very few options for nlp with Bengali language detection. I will try to add couple of hands on example on how well spark nlp can handle Bengali language, so that their claim on multi language support is truly verified.

My review will consist mostly three segments, they are we can say three basic pillars I will constructing my review.

- How easy it is to setup and get it going for first time user
- How efficiently it can do basic nlp tasks like tokenization, stop words removal, pos tagging
- Multi Language support

I will use Python as choice of programming language. IDE will be VS code.

Java version will be 1.8, Python 3.6, PySpark 2.4.4

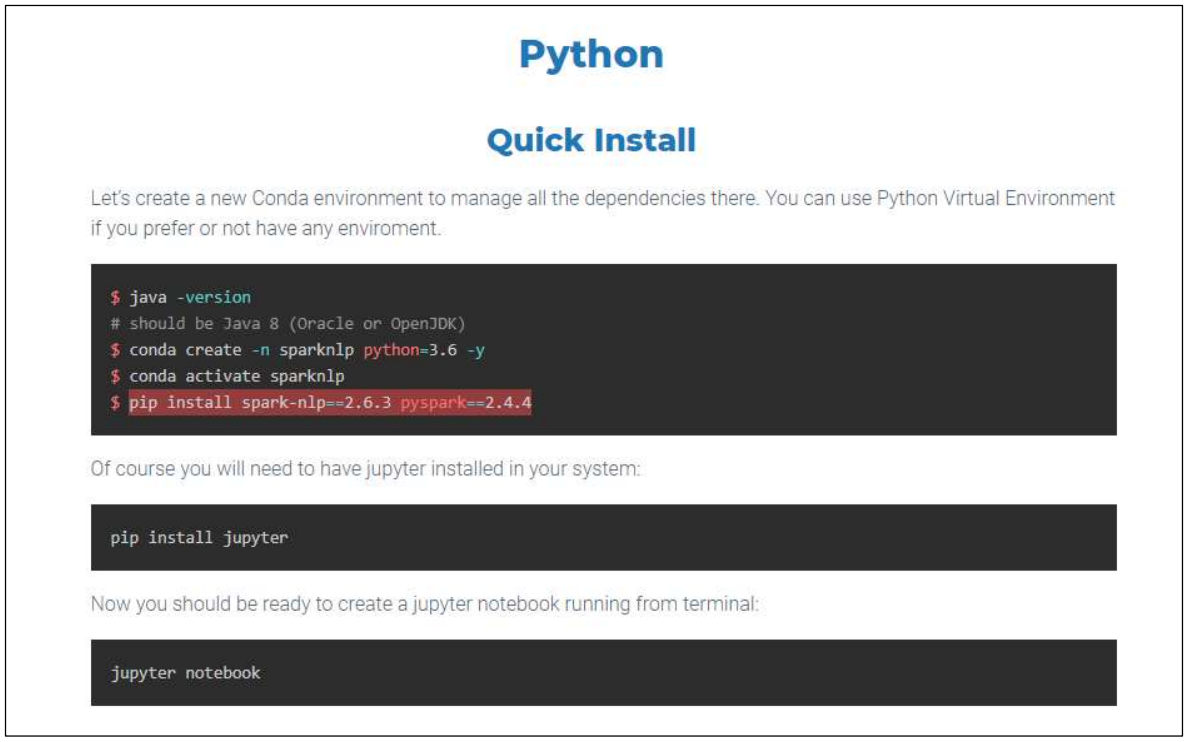
Most of my review I will be adding code snippet, results which will be taken from my local machine setup and logs.

I will not cover deep text analysis or complex language models.

I will list down any difficulties I faced while achieving some aspects of nlp.

Setup

I started to do a local setup on my windows machine from provided instructions from product homepage.



Python

Quick Install

Let's create a new Conda environment to manage all the dependencies there. You can use Python Virtual Environment if you prefer or not have any enviroment.

```
$ java -version
# should be Java 8 (Oracle or OpenJDK)
$ conda create -n sparknlp python=3.6 -y
$ conda activate sparknlp
$ pip install spark-nlp==2.6.3 pyspark==2.4.4
```

Of course you will need to have jupyter installed in your system:

```
pip install jupyter
```

Now you should be ready to create a jupyter notebook running from terminal:

```
jupyter notebook
```

Up until installation it was good, I was using vs code, I wrote basic import code, stared spark nlp, it got started successfully. Problem occurred when I tried to download one of their pre trained model using below command:

```
pipeline = PretrainedPipeline('explain_document_dl', lang='en')
```

It gave me an error below,

```
explain_document_dl download started this may take some time. Approx size to downloa
d 168.4 MB [OK!]
--------------------------------------------------------------------------- Py4JJava
Error Traceback (most recent call last) ~\anaconda3\lib\site-packages\pyspark\sql\ut
ils.py in deco(*a, **kw) 62 try: ---> 63 return f(*a, **kw) 64 except py4j.protocol.
Py4JJavaError as e: ~\anaconda3\lib\site-packages\py4j\protocol.py in get_return_val
ue(answer, gateway_client, target_id, name) 327 "An error occurred while calling {0}
{1}{2}.\n". --> 328 format(target_id, ".", name), value) 329 else: Py4JJavaError: An
error occurred while calling z:com.johnsnowlabs.nlp.pretrained.PythonResourceDownloa
der.downloadPipeline. : java.lang.IllegalArgumentException: requirement failed: Was
not found appropriate resource to download for request: ResourceRequest(explain_docu
ment_dl,Some(en),public/models,2.6.3,2.4.4) with downloader: com.johnsnowlabs.nlp.pr
etrained.S3ResourceDownloader@dd90bca
```

Next step I tried to find an alternative by downloading their pre-trained model offline from below link,

https://nlp.johnsnowlabs.com/docs/en/models

and tried to load them manually using below

```
PretrainedPipeline.from_disk('/root/cache_pretrained/explain_document_ml_en_2.4.0_2.4_1580252705962')
```

But that also gave me different sets of error,

```
Py4JJavaError Traceback (most recent call last) <ipython-input-20-bd49dee02dfa> in <module> ----> 1 pipeline = NerDLModel.load('C:/Users/dipta/cache_pretrained/ner_dl_en_2.4.3_2.4_1584624950746') ~\anaconda3\lib\site-packages\pyspark\ml\util.py in load(cls, path) 360 def load(cls, path): 361 """Reads an ML instance from the input path, a shortcut of `read().load(path)`.""" --> 362 return cls.read().load(path) 363 364 ~\anaconda3\lib\site-packages\pyspark\ml\util.py in load(self, path) 298 if not isinstance(path, basestring): 299 raise TypeError("path should be a basestring, got type %s" % type(path)) --> 300 java_obj = self._jread.load(path) 301 if not hasattr(self._clazz, "_from_java"): 302 raise NotImplementedError("This Java ML type cannot be loaded into Python currently: %r" ~\anaconda3\lib\site-packages\py4j\java_gateway.py in __call__(self, *args) 1255 answer = self.gateway_client.send_command(command) 1256 return_value = get_return_value( -> 1257 answer, self.gateway_client, self.target_id, self.name) 1258 1259 for temp_arg in temp_args:
```

After spending hours on debugging, I was not able to resolve the issue, below are things I have tried

- Using different vendor Java provider, I was using grallvm, changed to openjdk and conda jdk
- Changing Spark and nlp version
- Changing directory of default offline models

None worked so far, and this is a common issue for Spark nlp according to numerous git issue I have checked, all boiled down to this spark or Hadoop does not play well with Windows , Linux or Mac OS is preferred. There is not much documentation on if there is any alternative, but according to a stackoverflow post, with new version it should be working.

https://stackoverflow.com/questions/57610129/do-spark-nlp-pretrained-pipelines-only-work-on-linux-systems

I have reached out to that contributor for my issue, yet to hear back from him.

Upon my unsuccessful attempt to make it work in Windows machine, I have switched to Google Colab, and set up my environment there, surprisingly everything went super smooth and I was able to create and setup my pipeline in no time. Below are the commands:

```
import os

! apt-get update -qq

! apt-get install -y openjdk-8-jdk-headless -qq > /dev/nul

os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"

os.environ["PATH"] = os.environ["JAVA_HOME"] + "/bin:" + os.environ["PATH"]

! java -version

! pip install --ignore-installed -q pyspark==2.4.4

! pip install --ignore-installed -q spark-nlp==2.6.3-rc1

import sparknlp

spark = sparknlp.start()

# params =>> gpu=False, spark23=False (start with spark 2.3)

print("Spark NLP version", sparknlp.version())

print("Apache Spark version:", spark.version)

from sparknlp.pretrained import PretrainedPipeline

pipeline = PretrainedPipeline('explain_document_ml', lang='en')
```

Conclusion

For now I would say they should create a separate segment of installation on different os in their home page, while I have not tried myself, I believe Linux or Mac would be obvious choice, if any one is trying setup for first time. But Google Colab would be best fit if one is trying to easy time saving setup that would not take hours of googling the common set up issues.

Noteworthy before we move on to testing NLP

Everything is spark nlp works through pipeline, we can think of as analyzers in Meta but with bigger capabilities. For performing a task you will be needing different sets of pre trained models, some for say tokenizing, stemming, stop words removal, we need set them with our own data, and run them through pipeline like a chain event. One should go through below basic concepts on how spark nlp annotator, transformers and pipelines work all together.

https://nlp.johnsnowlabs.com/docs/en/concepts

These concepts are very well documented and is very easy to understand. Also, another noteworthy point, as it is backed by Apache spark, all of spark capabilities are available to use. I found use of spark dataframe and support is very useful.

NLP in Action

I have tried some basic NLP action using spark nlp,

First test document I used is below,

```
testDoc = '''
Learning Natural Langage Processing is fuf. John Wick is enjoying nlp.
'''
```

Using any of the model is very straight forward process, you choose your pre trained model from below link,

https://nlp.johnsnowlabs.com/models

and use them to download in local and they are ready to use like below

```
[11] pipeline_local = PretrainedPipeline.from_disk('/root/cache_pretrained/explain_document_ml_en_2.4.0_2.4_1580252705962')

[17] %%time

     result = pipeline.annotate(testDoc)

     CPU times: user 28.9 ms, sys: 11.9 ms, total: 40.8 ms
     Wall time: 114 ms

[18] result.keys()

     dict_keys(['document', 'spell', 'pos', 'lemmas', 'token', 'stems', 'sentence'])
```

All the model comes with basic capabilities, all of them well documented you can go to individual model and see what they can do with example. You list them using keys keyword that will show what that model is trained to do, like our above example can do spell check, pos tagging etc.

Only time taking process is downloading the model for first time, after they are downloaded everything is blazing fast, I have not tested them with bigger documents. But with similar sets what we have used MP1 they are showing impressing results.

Example of tokenization POS tagging and Stemming

```
[22] list(zip(result['token'], result['lemmas'], result['stems'], result['spell']))

    [('Learning', 'Learning', 'learn', 'Learning'),
     ('Natural', 'Natural', 'natur', 'Natural'),
     ('Langage', 'Lange', 'lang', 'Lange'),
     ('Processing', 'Processing', 'process', 'Processing'),
     ('is', 'be', 'i', 'is'),
     ('fuf', 'fuf', 'fuf', 'fuf'),
     ('.', '.', '.', '.'),
     ('John', 'John', 'john', 'John'),
     ('Wick', 'Wick', 'wick', 'Wick'),
     ('is', 'be', 'i', 'is'),
     ('enjoying', 'enjoy', 'enjoi', 'enjoying'),
     ('nlp', 'nlp', 'nlp', 'nlp'),
     ('.', '.', '.', '.')]
```

They have in built panda dataframe support, that is very useful, example below

```
[23] import pandas as pd
     df = pd.DataFrame({'token':result['token'],
                        'corrected':result['spell'], 'POS':result['pos'],
                        'lemmas':result['lemmas'], 'stems':result['stems']})
     df
```

	token	corrected	POS	lemmas	stems
0	Learning	Learning	NNP	Learning	learn
1	Natural	Natural	NNP	Natural	natur
2	Langage	Lange	NNP	Lange	lang
3	Processing	Processing	NNP	Processing	process
4	is	is	VBZ	be	i
5	fuf	fuf	NN	fuf	fuf
6	.	.	.	.	.
7	John	John	NNP	John	john
8	Wick	Wick	NNP	Wick	wick
9	is	is	VBZ	be	i
10	enjoying	enjoying	VBG	enjoy	enjoi
11	nlp	nlp	NN	nlp	nlp
12	.	.	.	.	.

Basic spell checking is also very easy, just need to add a new model, see example below.

```
[31] spell_checker_dl = PretrainedPipeline('check_spelling_dl', lang='en')
```

```
check_spelling_dl download started this may take some time.
Approx size to download 112.1 MB
[OK!]
```

```
[35] text = 'John Wick loves hif mustang'

     result = spell_checker_dl.annotate(text)

     list(zip(result['token'], result['checked']))
```

```
[('John', 'John'),
 ('Wick', 'Wick'),
 ('loves', 'loves'),
 ('hif', 'his'),
 ('mustang', 'Mustang')]
```

If you notice above his was spelled wrong, it was corrected.

Multi sentence line detection is very intuitive, I used below text from Macbeth Act III

```
['Macbeth becomes King of Scotland but is plagued by feelings of insecurity. He
remembers the prophecy that Banquos descendants will inherit the throne and
arranges for Banquo and his son Fleance to be killed. In the darkness, Banquo is
murdered, but his son escapes the assassins. At his state banquet that night,
Macbeth sees the ghost of Banquo and worries the courtiers with his mad
response. Lady Macbeth dismisses the court and unsuccessfully tries to calm her
husband.']
```

```
[36] doc_list = ['Macbeth becomes King of Scotland but is plagued by feelings of insecurity. He remembers the prophecy that Banquos d

     doc_list
```

```
['Macbeth becomes King of Scotland but is plagued by feelings of insecurity. He remembers the prophecy that Banquos descendants
```

```
[37] pipeline = PretrainedPipeline('explain_document_ml', lang='en')
```

```
explain_document_ml download started this may take some time.
Approx size to download 9.4 MB
[OK!]
```

```
result_list = pipeline.annotate(doc_list)
```

Analyzing above document it will automatically detect multi sentence and put them on Result set array. Like below

```
result_list[0]
```

```
{'document': ['Macbeth becomes King of Scotland but is plagued by feelings of insecurity. He remembers the pr
 'lemmas': ['Macbeth',
  'become',
  'King',
  'of',
  'Scotland',
  'but',
  'be',
  'plague',
  'by',
  'feelings',
  'of',
  'insecurity',
  '.',
  'He',
  'remember',
  'the',
  'prophecy',
  'that',
  'Banquo',
  'descendant',
  'will',
  'inherit',
  'the',
  'throne',
```

You can also get a detailed result from your document very easily, like below.

```
[42] detailed_result = pipeline.fullAnnotate('John Wick checked into Continental & talked to manager Winston')
```

```
[43] detailed_result
```

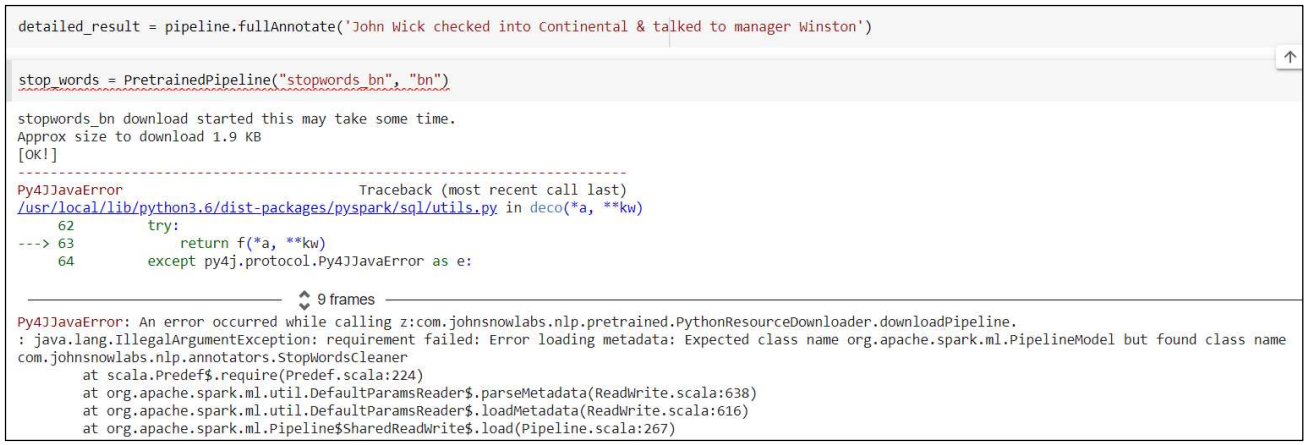
```
[{'document': [Annotation(document, 0, 61, John Wick checked into Continental & talked to manager Winston, {})],
  'lemmas': [Annotation(token, 0, 3, John, {'confidence': '1.0', 'sentence': '0'}),
   Annotation(token, 5, 8, Wick, {'confidence': '1.0', 'sentence': '0'}),
   Annotation(token, 10, 16, check, {'confidence': '1.0', 'sentence': '0'}),
   Annotation(token, 18, 21, into, {'confidence': '1.0', 'sentence': '0'}),
   Annotation(token, 23, 33, Continental, {'confidence': '1.0', 'sentence': '0'}),
   Annotation(token, 35, 35, &, {'confidence': '0.0', 'sentence': '0'}),
   Annotation(token, 37, 42, talk, {'confidence': '1.0', 'sentence': '0'}),
   Annotation(token, 44, 45, to, {'confidence': '1.0', 'sentence': '0'}),
   Annotation(token, 47, 53, manager, {'confidence': '1.0', 'sentence': '0'}),
   Annotation(token, 55, 61, Winston, {'confidence': '1.0', 'sentence': '0'})],
  'pos': [Annotation(pos, 0, 3, NNP, {'word': 'John'}),
   Annotation(pos, 5, 8, NNP, {'word': 'Wick'}),
   Annotation(pos, 10, 16, VBD, {'word': 'checked'}),
   Annotation(pos, 18, 21, IN, {'word': 'into'}),
   Annotation(pos, 23, 33, NNP, {'word': 'Continental'}),
   Annotation(pos, 35, 35, CC, {'word': '&'}),
   Annotation(pos, 37, 42, VBD, {'word': 'talked'}),
   Annotation(pos, 44, 45, TO, {'word': 'to'}),
   Annotation(pos, 47, 53, NN, {'word': 'manager'}),
   Annotation(pos, 55, 61, NNP, {'word': 'Winston'})],
  'sentence': [Annotation(document, 0, 61, John Wick checked into Continental & talked to manager Winston, {'sentence': '0'})],
  'spell': [Annotation(token, 0, 3, John, {'confidence': '1.0', 'sentence': '0'}),
   Annotation(token, 5, 8, Wick, {'confidence': '1.0', 'sentence': '0'}),
   Annotation(token, 10, 16, checked, {'confidence': '1.0', 'sentence': '0'}),
   Annotation(token, 18, 21, into, {'confidence': '1.0', 'sentence': '0'}),
   Annotation(token, 23, 33, Continental, {'confidence': '1.0', 'sentence': '0'}),
   Annotation(token, 35, 35, &, {'confidence': '0.0', 'sentence': '0'}),
```

conclusion

I have not quite deep dived into it, but I have tested most of the basic functionality and It was impressive. What I liked most is their pre trained models, list is extensive and so many to choose from. Each are well documented and very easy to use. What ever we use is meta, they are all readily available and performance is superb. I have also tried use them in little, bigger document (Not included here) spell check on 100-line document took less than 2.5 secs. Also using pipeline and setting up a bigger pipeline is breeze, you can set up n number of nlp set, and set them in one pipeline was super easy. Also, it did not feel very power-hungry application with humble setup of Google colab most of action was without hiccups. I would say spark nlp has superb future ahead.

Multi Language Support

I tried downloading stop words cleaner for Bengali language, unfortunately it was not able to download the model, I used couple different way, but it was not working.



```
detailed_result = pipeline.fullAnnotate('John Wick checked into Continental & talked to manager Winston')

stop_words = PretrainedPipeline("stopwords_bn", "bn")

stopwords_bn download started this may take some time.
Approx size to download 1.9 KB
[OK!]
---------------------------------------------------------------------------
Py4JJavaError                             Traceback (most recent call last)
/usr/local/lib/python3.6/dist-packages/pyspark/sql/utils.py in deco(*a, **kw)
     62         try:
---> 63             return f(*a, **kw)
     64         except py4j.protocol.Py4JJavaError as e:

                                  9 frames
Py4JJavaError: An error occurred while calling z:com.johnsnowlabs.nlp.pretrained.PythonResourceDownloader.downloadPipeline.
: java.lang.IllegalArgumentException: requirement failed: Error loading metadata: Expected class name org.apache.spark.ml.PipelineModel but found class name
com.johnsnowlabs.nlp.annotators.StopWordsCleaner
        at scala.Predef$.require(Predef.scala:224)
        at org.apache.spark.ml.util.DefaultParamsReader$.parseMetadata(ReadWrite.scala:638)
        at org.apache.spark.ml.util.DefaultParamsReader$.loadMetadata(ReadWrite.scala:616)
        at org.apache.spark.ml.Pipeline$SharedReadWrite$.load(Pipeline.scala:267)
```

There were multiple example of this model, but somehow I was not able see that in action, I was quite exited about this part, as there is not much nlp product out there that supports Bengali, I raised a git issue for this, I am yet get a reply from them.

Conclusion

I am not in position to confirm if multi language support is not working, as there are hundreds of examples with many languages, there must be something I am missing here. I would wait for my git issues to resolve and make my final comments.

The End