Google Colab Link: https://colab.research.google.com/drive/1KKxJVdkO439aFkDLUCs7KuNmyurh0l9f?usp=sharing

LAB E4 Regular Expression

1. Import the necessary libraries.

In [24]:
```
import re
```

2. Load the text in labE4.txt to a variable text.

In [25]:
```
text = ""
data_path = "C:\spark\MCA\Semester1\E3_NLP\input\lab_e4\LabE4.txt"
with open(data_path) as f:
    text = f.read()
```

3. Apply regular expression and clean the data.

In [26]:
```
text = re.sub(r"\n\n", r"\n", text)
text = re.sub(r"-", "_", text)
```

In [27]:
```
pattern = '^([^0-9][^*_*][^\\nNOTE*][^\\nWEBVTT*][^\\n*][^0-9*]
[^*\\nNOTE*]*)'
list_valid_txt = re.findall(pattern, text, re.MULTILINE)
```

In [28]:
```
vtxt = ""
for i in list_valid_txt:
    vtxt = vtxt + i + " "
vtxt = vtxt.strip()
```

In [29]:
```
vtxt = re.sub("\s\S*_\S*\s", " ", vtxt)
```

In [30]:
```
print("Final Cleaned text = \n", vtxt)
```

```
Final Cleaned text =
 Hello and welcome back. In the previous video we have seen in jargon the key terms and
so on. In this video we are going to look into NLP data versus  What are the different
categories of NLP data and the data itself? There are two categories which is structure
d and unstructured, so we'll be looking into all these topics in this meeting. Let us s
ee the data in a day. There are a lot of data that is generated in a day, so there are
around 500 millions of reads, So all of which which is relevant to natural language pro
cessing. Let's see. What is data in LP? So what is data? Data that stores language data
quality data. Basically there are two categories. One is the textual data and the speec
h data. In this course we will be only building with textual data alone. So here is an
example of a text data because there is some text here and you have the banking data. T
here are a lot of columns. There are a lot of words in each of these columns, so you ca
n consider this to be a text data. Everything is organized  And let's see another examp
le. So here we have a news article on coronavirus so. When compared with the previous s
lide, which is the banking data, what differences did you see? Yes, well, it's not orga
```

nized. There is no specific format. You have just some heading and followed by some tex t, some random text and so on. So you have a data which is organized and you have a dat a that is not organized. What about this? Yes here also. Even though you don't see any kind of organization, but you can have like this, is this has a special meaning or hash tag and so on. So this can also be considered as an unstructured or unorganized data. W hat about this? This is the news classification data set, though it seems to be a kind of organized, but you look into each of the. Who's in this column? You find that they a re all unorganized, so these are all news. Is different news that are there and they ca tegories. OK, so this is even though this is a tabular data. This is also unstructured data. So. So we have two categories. One is the structured data and the other is unstru ctured data in structured data fixed dimension, just as the banking data set that we sa w, everything is organized usually in tables or in a key value pair like Jason. Object JS And it is simple to process because we know well in hand what are we different? Froz en columns that are there in the daytime so. It is processed directly and there is no n eed for intermediate representations. Brothers in unstructured data. There is no specif ic dimension, so it's very hard and it is not organized and also there is no specific o rder thus making processing very complex and it needs lot of intermediate representatio n which we will have to study. How to learn. Converting the unstructured into some kind of structured format. So here is a task for you. There are there are. These are the res ources. I'll text data or some kind of data so you have to think you have to just categ orize them, whether structure or unstructured. Yes, so we have news article. Yes, this is unstructured. You have visuals from the CC Yes it is also unstructured. What about t he data here? Yes, there is some format, so it can be considered as. A structured data? What about this? This is a visual or this is the audio, so it is also unstructured. Wha t about this being Jason object? He was a Barracuda is not just so they are in key valu e pairs or disease structure data. What about meets and the post and the social media. Yes, you can write anything and the length can be. There is no fixed dimension so this is also what about this web page? so webpage where there is some organization without H by the body tag or baby Dev Dag. And maybe there is some format, but if you go into eac h of the tags you can find lot of unstructured data. So you can consider this to be a s ingle structure because overall it is structured. But when you go into it there you fin d lot of unstructured ones. So when you see the comparison of the structured versus uns tructured, you can see the explosion of unstructured a lot. You can see there are lot o f data that is being generated every year and the ratio in which structured and unstruc tured is very much. So you have lot of unstructured data when compared to structured da ta. So a lot of processing and a lot of work has to be done when we're dealing with uns tructured data. So in this video we have seen what is data and that we'll be focusing o nly on the text data and most of the text data. You can have their unstructured, so uns tructured means there is no specific formula. We can take the example of the news artic le, so most of the data is unstructured when compared with the structured data and next we will start with another pipeline and energy. I put this thank you.

In [31]:
```
text = ""
with open(data_path) as f:
    text = f.read()
```

In [32]:
```
def match_pattern(pattern, text, question_pattern):
    m = re.search(pattern, text, re.MULTILINE)
    print(m)
    if(m):
        print("Match Found for pattern: ", question_pattern)
    else:
        print("No match found for pattern: ", question_pattern)
```

In [33]:
```
match_pattern("[0-9]{2}\-[0-9]{2}\-[0-9]{4}", text, "MM-DD-YYYY")
```

```
None
No match found for pattern:  MM-DD-YYYY
```

In [34]:
```
match_pattern("[0-9]{2}:[0-9]{2}\:[0-9]{2}", text, "HH:MM:SS")
```

```
<re.Match object; span=(23, 31), match='00:06:54'>
Match Found for pattern:  HH:MM:SS
```

In [35]:
```
match_pattern("[0-9]{2}:[0-9]{2}\:[0-9]{2} [AM|PM]", "00:06:54 PM",
"HH:MM:SS AM or HH:MM:SS PM")
```

```
<re.Match object; span=(0, 10), match='00:06:54 P'>
Match Found for pattern:  HH:MM:SS AM or HH:MM:SS PM
```

In [36]:
```
match_pattern("^M(r.|s.|rs.) [A-Za-z]*", "Mr. Manav", "Mr. .... or Ms. ....
or Mrs. ....")
```

```
<re.Match object; span=(0, 9), match='Mr. Manav'>
Match Found for pattern:  Mr. .... or Ms. .... or Mrs. ....
```

In [37]:
```
match_pattern("^AA.SC.P2MCA[0-9]{7}$", "AA.SC.P2MCA2107423", "Enrollment
Number: AA.SC.P2MCA2107423")
```

```
<re.Match object; span=(0, 18), match='AA.SC.P2MCA2107423'>
Match Found for pattern:  Enrollment Number: AA.SC.P2MCA2107423
```

In [38]:
```
match_pattern("^([0-9]{4}-[0-9]{7}|[0-9]{3}-[0-9]{8})", "022-12345678",
"Indian Landline Number: 0476-2802017 or 022-12345678")
```

```
<re.Match object; span=(0, 12), match='022-12345678'>
Match Found for pattern:  Indian Landline Number: 0476-2802017 or 022-12345678
```

In [39]:
```
match_pattern("^\+[0-9]{2}-[0-9]{10}", "+12-3456789000", "Any Mobile
Numbers: +91-1234567890. +12-3456789000")
```

```
<re.Match object; span=(0, 14), match='+12-3456789000'>
Match Found for pattern:  Any Mobile Numbers: +91-1234567890. +12-3456789000
```

In [40]:
```
match_pattern("^[0-9]{3} [0-9]{3}", "700 135", "Indian Postal Pin code: 690
525")
```

```
<re.Match object; span=(0, 7), match='700 135'>
Match Found for pattern:  Indian Postal Pin code: 690 525
```

In [41]:
```
#"Tokenize a sentence (?!.)"
list_tkn = re.split("(\?|\!|\.)", text)
list_tkn[:15]
```

Out[41]:
```
['WEBVTT\n\nNOTE duration:"00:06:54"\n\nNOTE recognizability:0',
 '.',
 '684\n\nNOTE language:en-us\n\nNOTE Confidence: 0',
 '.',
 '62458827231579\n\n76f86f8d-0f62-433a-a8d4-7b7088fc7070\n00:00:07',
 '.',
 '110 --> 00:00:08',
 '.',
 '730\nHello and welcome back',
 '.',
 '\n\nNOTE Confidence: 0',
 '.',
 '62458827231579\n\n3973772b-485a-4b33-8be2-f9f90b42089e\n00:00:08',
 '.',
 '730 --> 00:00:11']
```

In [42]:
```
match_pattern("[0-9]{4}", "700 135 3557", "finding any 4 digit in a text")
```

```
<re.Match object; span=(8, 12), match='3557'>
Match Found for pattern:  finding any 4 digit in a text
```

In [43]:
```
match_pattern("^[0-9]{4}-[0-9]{4}-[0-9]{4}-[0-9]{4}$",
"7008-1358-3557-9754", "Card Number :(OOOO-OOOO-OOOO-OOOO)")
```

```
<re.Match object; span=(0, 19), match='7008-1358-3557-9754'>
Match Found for pattern:  Card Number :(OOOO-OOOO-OOOO-OOOO)
```

In [44]:
```
match_pattern("^@[A-z0-9_]{5,15}$", "@stardust_df", "Twitter User :
(@username)")
```

```
<re.Match object; span=(0, 12), match='@stardust_df'>
Match Found for pattern:  Twitter User : (@username)
```

In [45]:
```
ipv4 = '''^(25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(
            25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(
            25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(
            25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)$'''
match_pattern(ipv4, "192.168.4.164", "IP Address")
```

```
<re.Match object; span=(0, 13), match='192.168.4.164'>
Match Found for pattern:  IP Address
```

In [46]:
```
ipv6 = '''(([0-9a-fA-F]{1,4}:){7,7}[0-9a-fA-F]{1,4}|
        ([0-9a-fA-F]{1,4}:){1,7}:|([0-9a-fA-F]{1,4}:)
        {1,6}:[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1
        ,5}(:[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}
        :){1,4}(:[0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{
        1,4}:){1,3}(:[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA
        -F]{1,4}:){1,2}(:[0-9a-fA-F]{1,4}){1,5}|[0-9a
        -fA-F]{1,4}:((:[0-9a-fA-F]{1,4}){1,6})|:((:[0
        -9a-fA-F]{1,4}){1,7}|:)|fe80:(:[0-9a-fA-F]{0,
        4}){0,4}%[0-9a-zA-Z]{1,}|::(ffff(:0{1,4}){0,1}
        :){0,1}((25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9
        ])\.){3,3}(25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0
        -9])|([0-9a-fA-F]{1,4}:){1,4}:((25[0-5]|(2[0-4]
        |1{0,1}[0-9]){0,1}[0-9])\.){3,3}(25[0-5]|(2[0-4]
        |1{0,1}[0-9]){0,1}[0-9]))'''
match_pattern(ipv6, "2001:db8:3:4::192.0.2.33", "IPv6 Address")
```

```
<re.Match object; span=(0, 24), match='2001:db8:3:4::192.0.2.33'>
Match Found for pattern:  IPv6 Address
```