VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

FALL SEMESTER 2020-2021

CSE4022 : Natural Language Processing

DIGITAL ASSIGNMENT-1

TOPIC:-

Faculty:- Prof.Sharmila Banu K

Submitted by:-

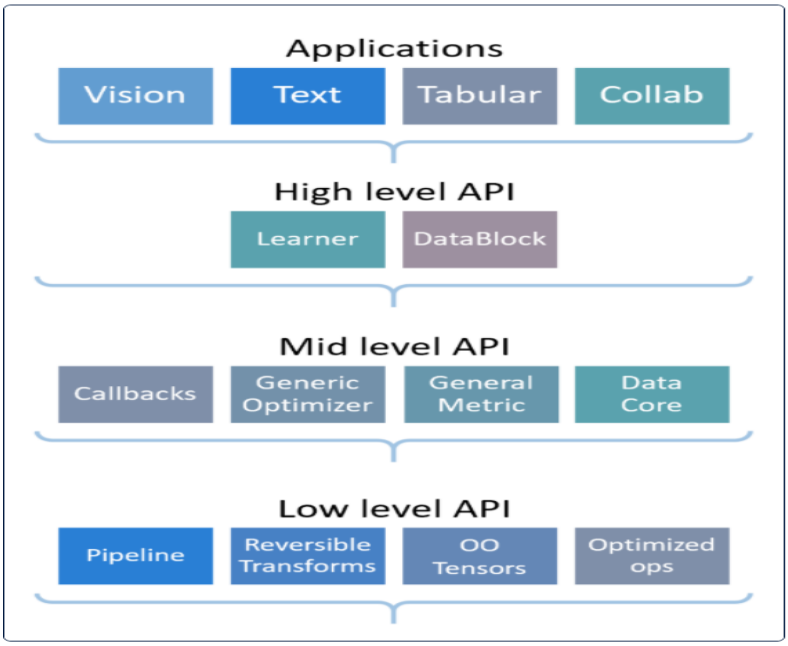
Shivam Anand (18BCE0490)
Slot:-E2+TE2

Introduction:-

Fastai is a deep learning library which provides practices with high-level component which also provides with quick and easy state of art that results quickly and efficiently in standard deep learning domain. It also provides researchers with low level components which are used to come up with new approach. Fastai aims to do both things i.e substantial compromise in case of use and it also enhances the performance .This is possible because of a carefully layered architecture, which provides information about the patterns of many deep learning and data processing techniques as far as decoupled abstractions[1]. These abstractions can be communicated concisely and transparently by utilising the dynamism of the underlying Python language and the adaptability of Pytorch library.
Fastai is organised and produced around two main design goals : to be productive and rapidly approachable, while likewise being profoundly hackable and configurable. It is based hierarchy of lower level APIs providing composable building blocks. Using the given properties the user wanting to rewrite part of high-level API or add new features and behaviour doesn't have to learn the building and functionalities of lowest level[1].

Some features of Fastai[2] :-

- It provides a new type dispatch system for Python along with a semantic type hierarchy for tensors.
- A GPU-optimized computer vision library which can be extended in pure Python.
- It allows the optimization algorithms to be implemented in 4-5 lines of code.
- It can access any part of the data, model, or optimizer and change it at any point during training
- It provides a new data block API.



Layered API of fastai

Code snippets :-

The following are rules applied to texts before it's tokenized.

`replace_all_caps(t)`

It replaces all the tokens into their lowercase.

```
test_eq(replace_all_caps("I'M SHOUTING"), f"{TK_UP} i'm {TK_UP} shouting")
test_eq(replace_all_caps("I'm speaking normally"), "I'm speaking normally")
test_eq(replace_all_caps("I am speaking normally"), "i am speaking normally")
```

`spec_add_spaces(t)`

It adds spaces in the text.

```
test_eq(spec_add_spaces('#fastai'), ' # fastai')
test_eq(spec_add_spaces('/fastai'), ' / fastai')
test_eq(spec_add_spaces('\\fastai'), ' \\ fastai')
```

Tokenizer[3]:-

Tokenization is the process of converting a sensitive piece of data into a random string of characters called a token that has no meaningful value if breached. After text pre-processing tokenization is applied .SpacyTokenizer is one of the most commonly used tokenizer.

```
tok = SpacyTokenizer()
inp,exp = "This isn't the easiest text.",["This", "is", "n't", "the", "easiest", "text", "."]
test_eq(L(tok([inp,inp])), [exp,exp])
```

Lemmatizer[3] :-

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item.

After tokenization lemmatizer is applied. Spacy lemmatizer is one of the most common used tokenizer,

```
from spacy.lemmatizer import Lemmatizer
lemmatizer = Lemmatizer()
[lemmatizer.lookup(word) for word in word_list]
```

Model Building[4]

Fastai uses houses various pre - trained language models that can be easily imported and used by changing and tuning the hyper parameters in accordance to the dataset given to us.

The below code snippet takes an example of how one such model can be used for sentiment analysis on the IMDB movie review data dataset.

```
df_imdb = pd.read_csv(path/'movies_data.csv')
df_imdb.head()
#Loading only few training and validation samples, for quick training time
trn_texts = df_imdb.loc[10000:14999, 'review'].values
trn_labels = df_imdb.loc[10000:14999, 'sentiment'].values
val_texts = df_imdb.loc[36000:38999, 'review'].values
val_labels = df_imdb.loc[36000:38999, 'sentiment'].values
np.random.seed(42)
trn_idx = np.random.permutation(len(trn_texts))
val_idx = np.random.permutation(len(val_texts))
trn_texts = trn_texts[trn_idx]
val_texts = val_texts[val_idx]
trn_labels = trn_labels[trn_idx]
val_labels = val_labels[val_idx]
col_names = ['labels','text']
df_trn = pd.DataFrame({'text':trn_texts, 'labels':trn_labels}, columns=col_names)
df_val = pd.DataFrame({'text':val_texts, 'labels':val_labels}, columns=col_names)
```

Topic selection:-

The fastai library is an amazing resource. it is easy to get a fastai model up and running in only a few lines of code. I choose fastai because of its modularity, high level apis that implemented state of the art techniques, and innovations that reduce the need for tons of compute but with the same performance characteristics. Performance wise, fastai is considered to be faster than the other frameworks like tensorflow and Pytorch. FastAI was built on top of Pytorch and will have additional helper functions which are fast enough and produces pretty good results.
Also, FastAI focuses on all areas like NLP, Computer Vision, etc. which are often not included in its counterparts like pyTorch and keras.

Hence, I chose fastai as my topic because not only it is a new and upcoming framework, but it has also been gaining a lot of popularity between amateurs and practitioners alike.

References:-

[1] https://docs.fast.ai/

[2] https://www.fast.ai/2020/02/13/fastai-A-Layered-API-for-Deep-Learning/#:~:text=A%20GPU%2Doptimized%20computer%20vision,4%2D5%20lines%20of%20code

[3] https://docs.fast.ai/text.core

[4] https://towardsdatascience.com/machine-learning-text-classification-language-modelling-using-fast-ai-b1b334f2872d