Luis Chavez, lwchave2, 669863000

Technology Review

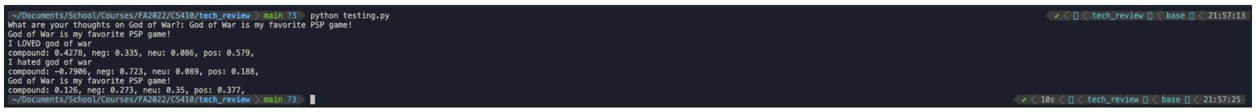
Overview of Sentiment Analysis Toolkits

Introduction

Natural Language Processing (NLP) is considered to have originated around the 1950s. Progress and development in this field has impacted the modern world in a plethora of ways, and in particular gauging public opinion. The importance of Sentiment Analysis is shown by the use of it in marketing campaigns, political campaigns, social media platform development, software development, public opinion polls, market predictions, etc. In the modern day there exists many NLP toolkits that include Sentiment Analysis or general purpose machine learning toolkits with NLP extensions. This review will focus on two such toolkits: (1) Natural Language Toolkit, NLTK and (2) TensorFlow. The aim is to identify the use case for each toolkit and provide a recommendation when it comes to Sentiment Analysis.

Body

NLTK is first and foremost a natural language software toolkit. However, the software offers a convenient machine learning extension. Through NLTK's scikitlearn module, support for classification algorithms such as SVMs, Naive Bayes, logistic regression, and decision trees is made possible. NLTK offers three methods of sentiment analysis each varying in complexity, ease of use, and customizability. The first method, which is the least involved, offers the user a general purpose sentiment analyzer through the nltk.sentiment.vader module. The user can input sentences and the analyzer will provide a review score (compound) broken down by three components (neg, neu, pos). The second method, is to train a classification algorithm via the nltk.sentiment.sentiment_analyzer module in a series of four steps: (1) choose classification algorithm, (2) partition collection of documents into training and test documents, (3) apply word features to document sets, and (4) train classifier on training documents and evaluate on test documents. The third and most involved method is to use the nltk.classify.scikitlearn module which wraps scikit-learn defined classifiers for use with NLTK libraries. In general, NLTK is low-level when it comes to sentiment-analysis, it is possible to define a very specific set of word features, word types (e.g. emoticons), etc to apply to any sentiment analysis or other type of categorization.



```
~/Documents/School/Courses/FA2022/CS410/tech_review > main ?3 > python testing.py          ✔ < tech_review < base < 21:57:13
What are your thoughts on God of War?: God of War is my favorite PSP game!
God of War is my favorite PSP game!
I LOVED god of war
compound: 0.4278, neg: 0.335, neu: 0.086, pos: 0.579,
I hated god of war
compound: -0.7906, neg: 0.723, neu: 0.089, pos: 0.188,
God of War is my favorite PSP game!
compound: 0.126, neg: 0.273, neu: 0.35, pos: 0.377,
~/Documents/School/Courses/FA2022/CS410/tech_review > main ?3 >          ✔ < 10s < tech_review < base < 21:57:25
```

Figure 1. Sample NLTK user program.

Google's TensorFlow is an end-to-end machine learning platform. As a consequence there is no 'out-of-the-box' sentiment analyzer, like there is in NLTK. Instead TensorFlow offers two methods by which to get a sentiment analyzer up and running. The first method for Tensor Flow is similar to NLTK's second method except that the classifier algorithms used are implicitly chosen when selecting a loss function from which to optimize. The series of steps are as follows: (1) choose loss function and appropriate optimizer (2) partition collection of documents into training and test documents and relevant data structures (3) standardize data (tokenize and vectorize documents) and (4) create neural network, train model and evaluate on test documents. The second method is similar to the first, but the distinction being the use of a BERT model for the sentiment analyzer. The series of steps are as follows: (1) choose loss function and appropriate optimizer (2) partition collection of documents into training and test documents and relevant data structures (3) load BERT model from TensorFlow Hub (Smart BERT recommended due to ease of fine-tuning) (4) use preprocessing BERT model to standardize data (tokenize and arrange data into tensors) and (5) train model and evaluate on test documents. TensorFlow classifier models can best be described using the figure below. TensorFlow clearly supports scalability, but it is not explicitly NLP accessible. For instance, in order to tune a classifier model to account for stemming and lemmatization, the preprocessing KerasLayer (see figure below) would need to be modified using external libraries.

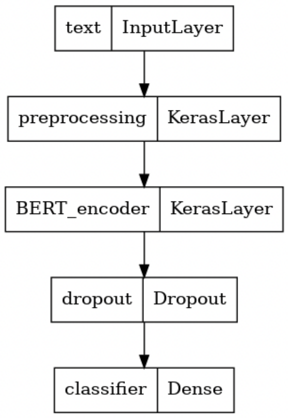


Figure 2. Sample TensorFlow classifier model.

Conclusion

For a small scale project with or without low-level needs NLTK is the superior toolkit due to: (1) ease of use, it is incredibly fast to get a sentiment analyzer up and running (2) NLP fine-tuning and (3) machine learning support, the scikitlearn module allows for machine learning if project in question needs it. For large scale projects with unique machine learning needs TensorFlow is recommended due to: (1) neural network options: RNNs, LSTMs, CNNs, and BERT (2) machine learning tools TensorFlow offer: optimizing for speed, size, and quality and (3) extensive documentation.

References

1. NLTK Sample usage for sentiment. https://www.nltk.org/howto/sentiment.html.
2. RNNs, LSTMs, CNNs, Transformers and BERT. https://medium.com/analytics-vidhya/rnns-lstms-cnns-transformers-and-bert-be003df3492b.
3. TensorFlow Basic text classification. https://www.tensorflow.org/tutorials/keras/text_classification.
4. TensorFlow Classify text with BERT. https://www.tensorflow.org/text/tutorials/classify_text_with_bert.