GITHUB

https://github.com/apurvmishra22/ProductAttributeExtraction

Product Attribute Extraction from Text

Introduction

Despite huge advancements in artificial intelligence, most businesses still manually extract product attributes from text. Because these algorithms only analyse numeric data, text is typically difficult to analyse with them. Word2vec and glove are just two examples of the many methods that have been developed to turn words into numerical vectors. In this project, we attempted to create a model that could extract certain features from a dataset's product description.

When attempting to extract product attributes from text, many difficulties arise. The text's layout and design are constantly changing, making it challenging to scrape data from a single algorithm because the model might not be able to keep up. With thousands of classes and millions of products to categorize, the extreme classification problem of product categorization using text data for eCommerce is very difficult. The characteristics themselves might be described differently, such as by using different spellings of the same brand name (GE and General Electric) or by using different unit measurements (7", 7", or 7 feet).

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.
Long short term memory networks, usually called LSTM – are a special kind of RNN. They were introduced to avoid the long-term dependency problem. In regular RNN, the problem frequently occurs when connecting previous information to new information.

Bidirectional long-short term memory(bi-lstm) is the process of making any neural network o have the sequence information in both directions backwards (future to past) or forward(past to future).
In bidirectional, our input flows in two directions, making a bi-lstm different from the regular LSTM. With the regular LSTM, we can make input flow in one direction, either backwards or forward. However, in bi-directional, we can make the input flow in both directions to preserve the future and the past information.

Dataset for my work is taken from this link https://raw.githubusercontent.com/maciej-cecot/brand-detection/master/models/lstm_brand_detection_model/helper_files/train.csv

Refer to this data for glove embedding: https://nlp.stanford.edu/projects/glove/
(unable to upload due to restriction of GitHub)
So, to use any data driven models to extract attributes from a sentence just like a product title, first we have to convert all these words to vectors. To convert the words we can use word2vec, Glove Embedding etc. Glove Embedding is used in this project. It is based on matrix factorization techniques on the word-context matrix. In word2vec, words are taken as training data to model a neural network and embedding captures whether words appear in similar context but in glove embedding, words co occurrence over the whole text is focused. 'Glove.6b.300d.txt' is used for converting words. Here, this glove embedding can map the words to vectors of length 300. For word embedding,first a zero matrix of size (number of words, embedding size) is created then find the word from the glove file and add the word vector to the matrix. For the prediction we use two deep learning model, Bi directional Long Short Term Memory (Bi-LSTM) and Convolutional Neural Network (CNN) Bi-LSTM is a sequence processing model, consisting of two LSTM layers one is forward and another one is backward. Bi-LSTM increases the amount of available information in the network. So, we can achieve high accuracy using Bi-LSTM. For our prediction model, 2 layers of Bi-LSTM model with 100 nodes per layer respectively are used. Optimized the hinge Loss using Nadam optimizer and iterated the model for 3 epochs. Bi- LSTM model summary, Another model we have tried to implement is Convolution Neural Network. CNN is generally used for image classification, but CNN can be applied for different NLP tasks. By varying the size of the kernels and layers, we can detect the pattern of the word vectors and we can predict our product attribute. To convert words to vectors in CNN model, we have written a separate code to create a word embedding file by taking around 1500 titles from the training dataset. I have used the first 1500 data for training from the dataset mentioned above as our word embedding file is created using these data. Used 4 convolution layers with ReLU activation function and 5 dense layers with 64 and 32 nodes per layer. Optimized the categorical cross entropy loss using stochastic descent gradient optimizer with learning rate 1e-5 and momentum 0.9 and used 5 epochs

Model Summary

1) Bi- LSTM model summary

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 21, 300)           6000300
_________________________________________________________________
bidirectional (Bidirectional (None, 21, 200)           320800
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               120400
_________________________________________________________________
dense (Dense)                (None, 5)                 505
=================================================================
Total params: 6,442,005
Trainable params: 6,442,005
Non-trainable params: 0
_________________________________________________________________
None
```

2) CNN Model Summary

```
Model: "model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 21)]              0
_________________________________________________________________
embedding_4 (Embedding)      (None, 21, 1000)          3667000
_________________________________________________________________
conv1d_16 (Conv1D)           (None, 19, 128)           384128
_________________________________________________________________
conv1d_17 (Conv1D)           (None, 17, 64)            24640
_________________________________________________________________
conv1d_18 (Conv1D)           (None, 15, 64)            12352
_________________________________________________________________
conv1d_19 (Conv1D)           (None, 13, 64)            12352
_________________________________________________________________
flatten_4 (Flatten)          (None, 832)               0
_________________________________________________________________
dense_29 (Dense)             (None, 64)                53312
_________________________________________________________________
dense_30 (Dense)             (None, 2)                 130
=================================================================
Total params: 4,153,914
Trainable params: 4,153,914
Non-trainable params: 0
_________________________________________________________________
None
```

A **CNN BiLSTM** is a hybrid bidirectional LSTM and CNN architecture. In the original formulation applied to named entity recognition, it learns both character-level and word-level features. The CNN component is used to induce the character-level features. For each word the model employs a convolution and a max pooling layer to extract a new feature vector from the per-character feature vectors such as character embeddings and (optionally) character type.

The Bi-LSTM and CNN model employed were able to predict some of the features from the unstructured text data. Some unwanted attributes were also predicted. With further augmentation in the model by optimizing the hyper-parameters this can be overcome.

In CNN model, optimizing hinge loss using Nadam, we are getting Hinge Loss as 0.8433 and 90.73% accuracy on training data and 0.8433 hinge loss and 89.75% accuracy on validation data after 3 iterations.
In Bi-LSTM model, optimizing hinge loss using Adam, 88.45% accuracy on training data was acheived