Abstract

During online interactions to purchase a product, the reviews left by customers become the new 'mouth of word' that sells the product. It is important to remove stop words, punctuation, special characters and stem the words in these reviews to reduce numbe r of features. We compare XgBoost and BERT model's performance to classify a free text as helpful to others.

Introduction

Classification of text based on a metric has been tackled several times as either a binary or multi class classification task. The state-of-the-art models perform better compared to tradition methods like XgBoost where these is no retention of temporal knowledge in the model. In XgBoost, the words are converted into numbers using either word embeddings of term frequency – inverse document frequency (tf-idf) and additional features are created using the modeler's knowledge and bias such as length of the sentence and pos tagging. With Seq2Seq model being able to store the temporal information the accuracy does increase but the complexity of the models rises exponentially and the loss of context with big sentences. BERT with its Transformers utilizes attention and creates parallelization inherent with its multi-head attention retaining context even in longer sentences.

Sampling

The dataset is highly imbalanced with the stats from 233.1 million data points as follows:

Type	Total	Count	Max	Min	Mean
Votes (0-5)	986334473.0	233055327	5	0	4.23
Helpful Votes	252098446.0	32591040	58206	0	7.73
Images	7064778	3914040	704	1	1.80

The votes per review on an average are 4.23, the data is heavily skewed towards positive votes. Similarly helpful votes and images are biased. However, we only use the helpful votes as the feature for modeling our classifier. It is assumed that the reviews without any values or empty are not helpful to others and have been assigned '0' class and all reviews with helpful votes above or equal to 1 have been assigned class '1'. To create train, valid and test dataset, a naïve approach of 50/50 helpful and non -helpful reviews ratio has been used for train and test sets.

Approaches

XgBoost

Feature Engineering

Data cleaning includes stemming, removing punctuations, special characters, stop words and lemmatization to reduce the feature space. The text is first tokenized, and cle aned for tf-idf which encodes the information about word occurrence and importance, LSA reduces the feature space to a fixed number of components to be fed into the model, which is n_components=300 for our model.

As the length of the review dictates how helpful a user might find it, we also include the length of each review as a second feature. This assumption comes from the understanding that a review which only has a few words would contain less information than a review with bigger length. However, this can be a bias as small reviews can be just as concise and information as a long one. Due to time constraints only two features are used to train models but features can be generated for reviews such as pos tagging.

Results

Test set metrics

- Accuracy=0.86

class	precision	recall	F1-score
0	0.87	0.98	0.92
1	0.63	0.20	0.30

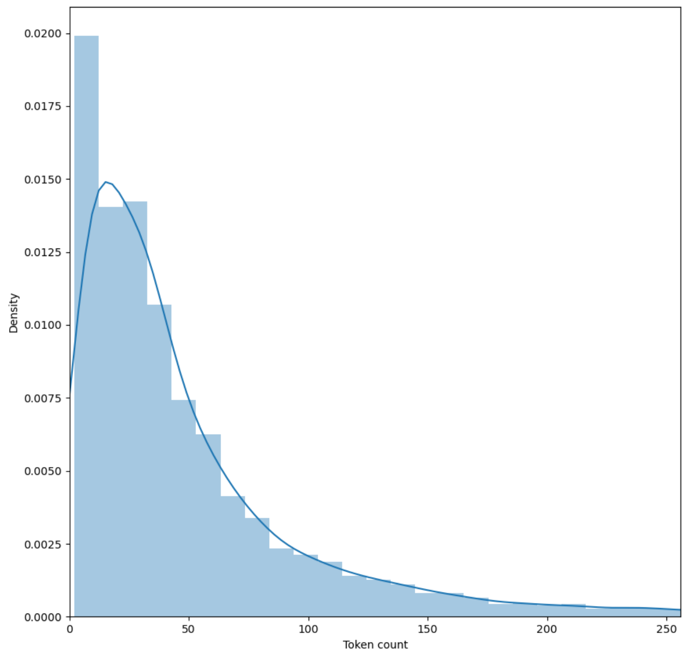
Conclusion

Performs bad on class ‘1’ with only 20% of the positive cases being recognized as positive and 30% being true positive predictions. Requires more training data and better sampling methods to improve precision.

BERT

Feature Engineering

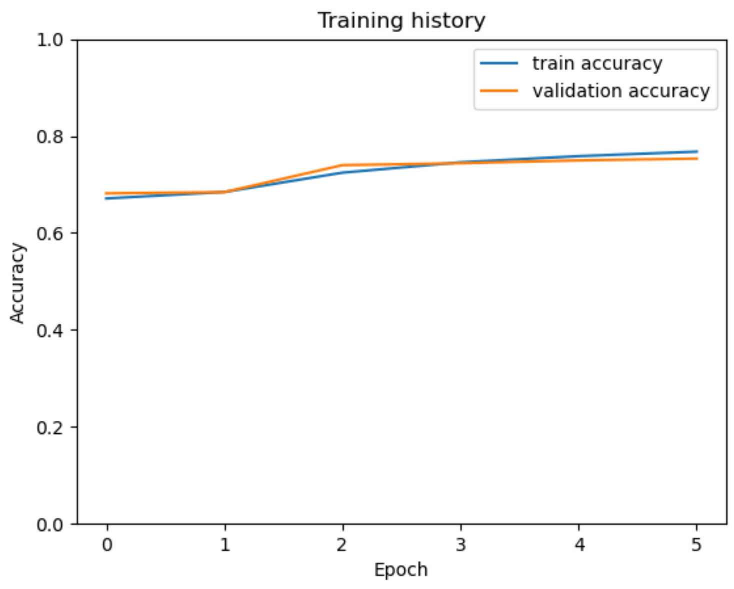
Selecting token size for BERT: From 100,00 randomly sampled data points using a uniform distribution method we find that most of the words have a maximum length of 150, to be conservative we select 200



Tokenization & Data Preprocessing: Utilize pre-train model ‘bert_base_cased’ and its tokenizer to tokenize and clean input data

Results

Epochs=6, Batch size=6. Model trains well with no clear signs of overfitting or underfitting.



Test set metrics

- Accuracy=0.75

class	precision	recall	F1-score
0	0.78	0.89	0.83
1	0.67	0.45	0.54

Conclusion

Generalizes well over test data, with decent f1-score for both classes. Performs much better than XgBoost but requires more data per epoch, number of epochs, more time to train and hyper parameter tunning

Future work

Sampling & selection bias

Instead of classifying each review as either 0 or 1, it would be interesting to see if normalized values of helpful votes can be used to break down into 4 categories of helpful, not helpful, somewhat helpful and very helpful.

In addition to randomness, also take into consideration the distribution of sentence length. So, for 50% of the data as helpful votes; it should include review with normally distributed sentence lengths and number of helpful votes. This would help remove selection bias in the data.

Languages for the reviews can be different than English. The model will perform badly on any language other than what it has been trained on, such reviews should either be considered outliers, converted to English or not included in the dataset.

Retraining & Hyper Parameter Tuning

Due to time constraints currently, the models have only been tested with one set of hyper parameters, which may not be the best parameters for the model given task. Grid search can be applied to optimize parameters. These include parameters inherent to the model as well as epoch and batch sizes.

Other ideas

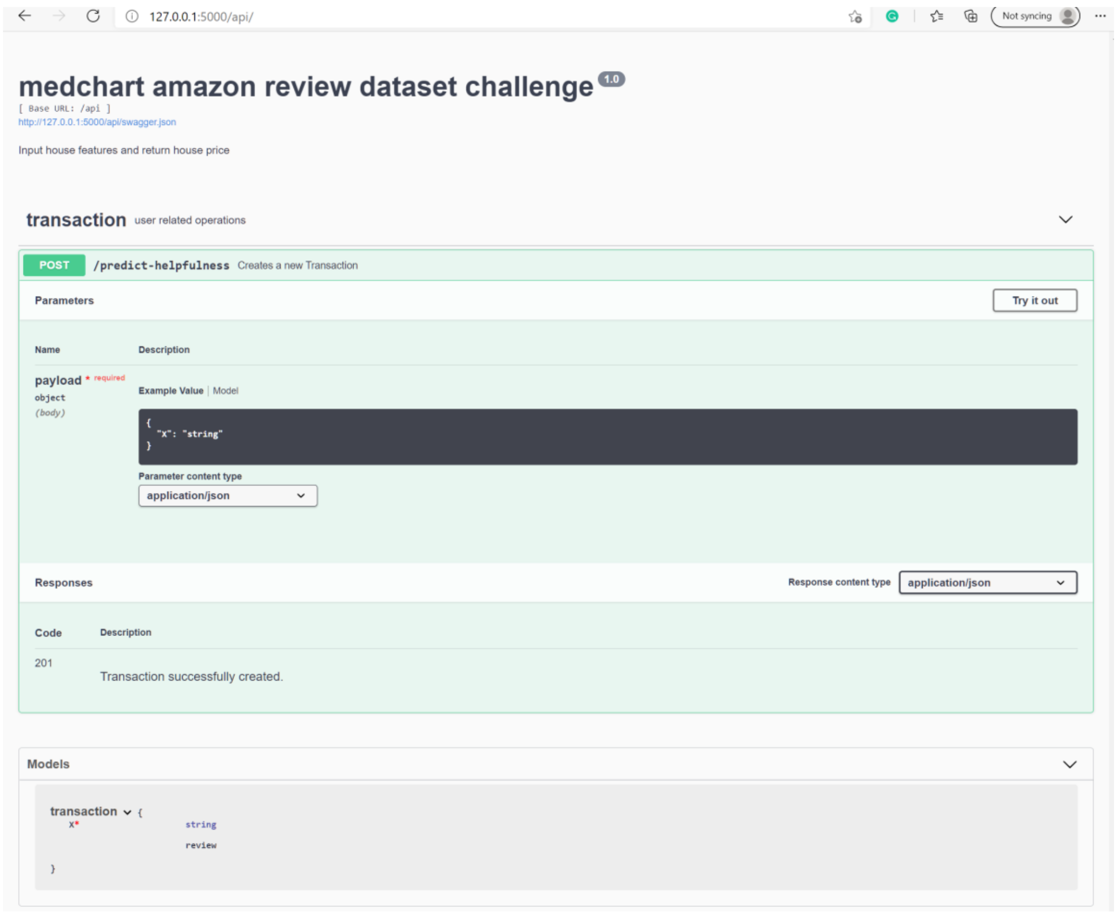
Ensemble: To train 2 models to predict vote and helpfulness as a regression metric, then combine the score to provide helpfulness ranking

Conclusion

The task of reviewing a free text consists of tackling data imbalance and sampling biases. Followed by preprocessing and tokenization of the text to be made useful for training models. The task is challenging and requires in-depth exploratory analysis to understand the review compositions and their features such as sentence lengths, helpful votes as described in the Future work section. XgBoost may be a viable approach and be less computationally expensive than BERT. However, BERT outperforms XgBoost on a given dataset because of its ability to retain context. The models have been deployed as a service using Flask-restx API.

API

Made using Flask-restx, provides UI using swagger.json. Accessible at domain/api/predict-helpfulness : accepts string as input and returns 'helpful' or 'not helpful'


127.0.0.1:5000/api/

medchart amazon review dataset challenge 1.0

[Base URL: /api]
http://127.0.0.1:5000/api/swagger.json

Input house features and return house price

transaction user related operations

POST /predict-helpfulness Creates a new Transaction

Parameters

Name	Description
payload * required object *(body)*	Example Value | Model

```
{
  "X": "string"
}
```

Parameter content type: application/json

Responses — Response content type: application/json

Code	Description
201	Transaction successfully created.

Models

```
transaction {
   X*                  string
                       review
}
```

Future Work

- Modularize model code into a class and add positive and negative unit tests for API and model
- Configure model to load on start-up and more configurations in `config.py`