**Project Report**

(Mid-May – Mid-July)

**Project Title:**

AI model for aspect-based multi-label text classification of customer reviews.

**Introduction:**

Classification of the negative sentimental customer reviews in the following categories:

|  |  |
| --- | --- |
| **L1 tags** | **L2 tags** |
| AC/Heater | Not effective, not present, not starting, noisy, |
| Comfort & Safety | Noise in the room, Bed uncomfortable, Door / window safety, Room lighting insufficient, Unsafe location, Privacy issue |
| Check-in Experience | Delay in check-in, Staff unfriendly, Incorrect booking details, Mentioned room category was not allotted, Staff not present at reception |
| Food Experience | Quantity was not enough, Variety was not enough, Food quality was not good, Food timing was not suitable, Dining area was small / dirty, Food was not available |
| Hotel Infrastructure | Seepage / paint decay on walls, Small size of room / washroom, Power backup / fluctuation issue, Broken tiles/floor |
| Hygiene & Cleanliness | Dirty/damaged linen and bedding, Overall, the hotel is not clean, Room not clean, Smell in room, Insects in the room |
| Room Equipment & Amenities | Missing facilities / amenities, Sockets & switches not working, Furniture broken/missing, Wardrobe & luggage rack, Room different from pictures |
| Staff & Service | Rude/Unfriendly staff, Slow service, Staff not available, Untrained staff |
| TV & Wi-Fi | TV unavailable / not working, TV channels inadequate, TV remote missing / not working, Wi-Fi not working, Wi-Fi slow, Wi-Fi password not given, Low data limit |
| Washroom | Hot water unavailable, Broken tap/shower/fixture, Smell / exhaust fan, Washroom not clean, Towel dirty / not provided, WC not clean / broken, Waste bin / toilet paper, Toiletries not present |

Developing an AI model will help to automate classification of reviews thus reducing manual labor. Further we can revert to concerned hotel’s manager regarding improving their reported facilities and hence it will result into increased ratings of our hotel.

To build an effective model, we need to represent documents such that it is understandable to machine learning classifier. Starting from very basic tfidf vectorizer and a general logistic regression for classification I switched to word embedding and neural networks for L1 and random forest for L2.

**Gathering Data:**

Approx. 19,000 annotated data set points collected from International Data was used for training the ML model.

**Data Pre-Processing:**

The collected data cannot be used directly for performing the analysis as there might be a lot of unorganized text data, extremely large text or noisy data. So, data pre-processing is considered an important step that helps in building machine learning models more accurately. It can be done in various steps.

**Data conversion:** Machine learning can handle numeric features. So, converting the target values into categorical features i.e. reduce the problem to binary classification.

**Ignoring the values:** Numeric values and punctuations are the cause of noise in our training data. Also pronouns and verbs are useless. Removing these will increase the accuracy of ML model. We can use inbuilt nltk corpus of python for removing stop words. It will take the sentence as input and replace them with a space. Also removing tabs and lines, results in improvement of accuracy.

**Replacing the values:** Lemmatization is the process of converting words to their stem while respecting their context.  The python module [nltk.stem](http://www.nltk.org/api/nltk.stem.html) contains a class called WordNetLemmatizer. In order to use it, one must provide both the word and its part-of-speech tag (adjective, noun, verb,) because lemmatization is highly dependent on context.

**Approach For L1**

**Feature Set Reduction:**

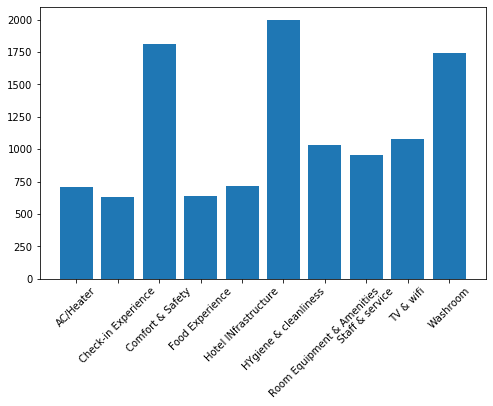
In this feature reduction technique, the original vector space is transformed to form a new minimalistic feature vector space. Original features are transformed into smaller set of transformed features. In our case, for text classification, this part is handled by Tokenizer API provided with keras. We need to pass maximum no. of features required.

**Word Embedding:**

Word Embeddingis a representation of text where words that have the same meaning have a similar representation. In other words it represents words in a coordinate system where related words, based on a corpus of relationships, are placed closer together.In the deep learning frameworks such as TensorFlow, Keras, this part is usually handled by an **embedding layer** which stores a lookup table to map the words represented by numeric indexes to their dense vector representations. Before passing to the embedding layer, text data is first encoded so that each word is first represented by a unique integer. This can be performed by using the tokenizer API provided with keras. We add padding to make all the vectors of same length (say max length).

**Data Distribution:**

The data distribution of each tag is shown in figure below

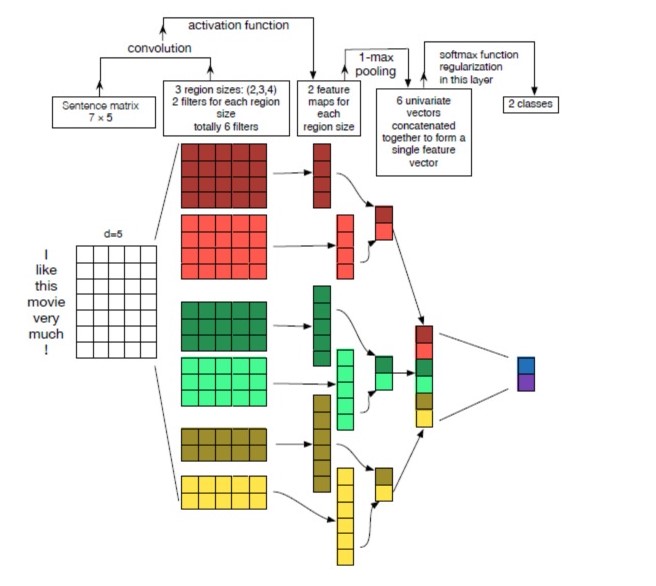
****

Thus, data set was highly imbalanced. Switching to binary classification by building 10 models for each tag in spite of using multi label classification.

**Model Building:**

There can be a lot of approaches like SVM, Random forest, Logistic Regression, Naïve Bayes, Neural Networks etc. for classification. But CNN (convolutional neural network) was seen to outperform them.

We begin with a tokenized sentence, which we convert to a sentence matrix. Let d be the dimension of the word vectors. Let each sentence is padded to length s. Then by treating each sentence matrix (dimension – s x d) as an image, we can perform convolutional on it via linear filters. We can vary the size as well as the number of filters to be used. Height of filter is referred as kernel size and number of filters are referred as output dimension. The dimensionality of the feature map generated by each filter will vary as a function of the sentence length and the kernel size. This is performed by using Conv1D module present with convolutional layer API provided with Keras. A pooling function is thus applied to each feature map to induce a fixed-length vector. A common strategy is 1-max pooling which extracts a scalar from each feature map. We can apply activation function to the dot product of filters and sentence matrix. Bias term can be added to reduce over-fitting. This is done by the dense layer provided with keras. A dropout layer stochastically “disables” a fraction of its neurons. This prevent neurons from co-adapting and forces them to learn individually useful features. The fraction of neurons we keep enabled is defined by the dropout\_keep\_prob input to our network. The network I built looks roughly is shown in figure below.

****

I have used 2 layered convolutional networks. At last use compilation layer to fix loss function and optimizer.

**Training & Testing:**

Hyper tune the parameters to get the best suited model and achieve maximum accuracy.

Now, divide the data set. 75% for training, 15% for testing and leave the rest for cross-validation. After few epochs, we will reach to the accuracy of approx. 90% for each of the 10 models.

**Results:**

The results of each model are as follows:

**AC/Heater**: The accuracy score is: 0.969989281886388

The roc\_auc\_score is: 0.9816903957528957

**Check-in Experience:** The accuracy score is: 0.9667738478027867

The roc\_auc\_score is: 0.9045868862099382

**Comfort & Safety:** The accuracy score is: 0.9346195069667739

The roc\_auc\_score is: 0.9204362879319319

**Food Experience:** The accuracy score is: 0.977491961414791

The roc\_auc\_score is: 0.9568310031886191

**Hotel Infrastructure:** The accuracy score is: 0.9678456591639871

The roc\_auc\_score is: 0.9213471283783784

**Hygiene & Cleanliness:** The accuracy score is: 0.9292604501607717

The roc\_auc\_score is: 0.9520265108500403

**Room Equipment & Amenities:** The accuracy score is: 0.9346195069667739

The roc\_auc\_score is: 0.9073407140634031

**Staff & Service:** The accuracy score is: 0.9506966773847803

The roc\_auc\_score is: 0.9180044663548645

**TV & Wi-Fi:** The accuracy score is: 0.9742765273311897

The roc\_auc\_score is: 0.9652322467448518

**Washroom:** The accuracy score is: 0.9260450160771704

The roc\_auc\_score is: 0.9352369380315917

**Approach For L2**

**Data Distribution:**

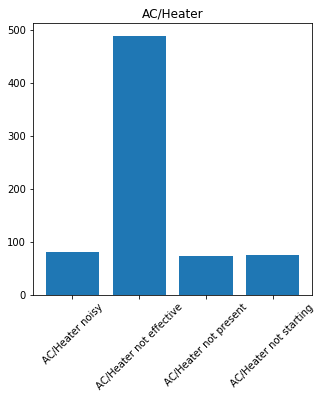
The data set of L2 is highly imbalanced with approx. 1k annotated data for each L1 tag. Predicting the accuracy score by using word embedding from tfidf and universal sentence encoder and training on Random Forest classifier, Multinomial naïve Bayes and kernel SVM, highest accuracy was achieved from tfidf vectorizer and Random Forest Classifier. Also we are here performing supervised learning and multiclass classification.

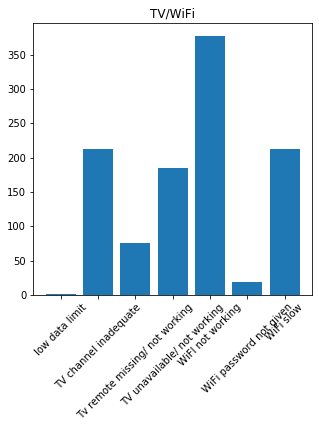
**Word Embedding:**

For L2 classification, I have used tfidf vectorizer. Tfidf stands for Term frequency - Inverse document frequency. It transforms the text into feature vectors that can be used as input estimator i.e. it creates a set of its own vocabulary from the entire set of documents. Term frequency summarizes how often a given word appears in a document. However Idf downscales words that appear a lot across documents. The [Tfidf Vectorizer](http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html) will tokenize documents, learn the vocabulary and inverse document frequency weightings, and allow you to encode new documents.

We have selected about 2k features i.e. our vocabulary will be of size 2k.

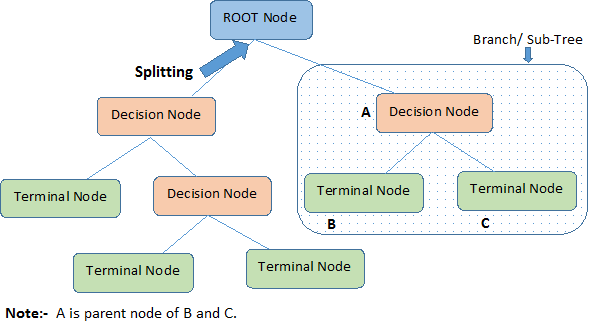
The data distribution is shown for few tags in following graphs:

`

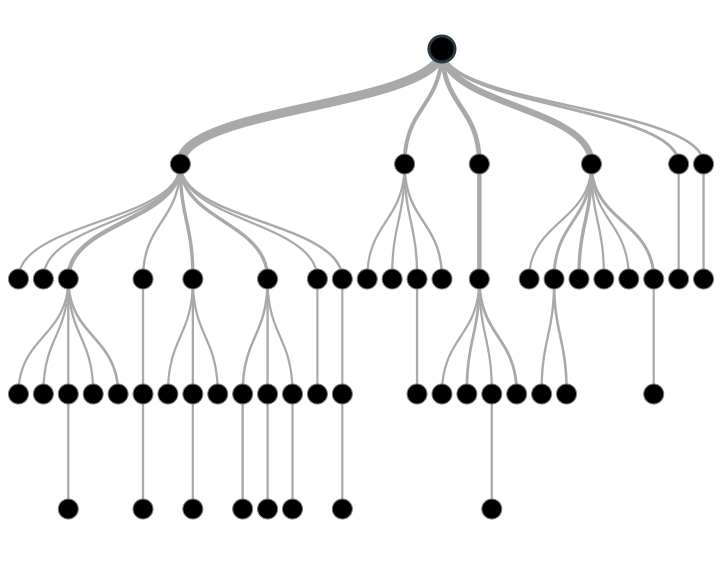


**Model Building:**

Due to discrepancy in data set, random forest was found suitable for training the model. We can use sklearn inbuilt library RandomForestClassifier to train the model. The working of this model is explained by the structure given below.



Random forest is a versatile machine learning method capable of performing both regression and classification. In this, we grow multiple trees. The tree is grown to the largest extent possible and there is no pruning. At the terminal node, we get the output of the model. Adjust the parameters to get the highest accuracy.



**Training & Testing:**

Split the data set into training set and test set, and check the accuracy using sklearn API. But due to discrepancy in the training data, use balanced accuracy score (average of recall value). We can change the probability distribution to get better results.

**Results:**

**AC/Heater**: The accuracy score is: 0.7482517482517482

**Check-in Experience:** The accuracy score is: 0.631578947368421

**Comfort & Safety:** The accuracy score is: 0.8125

**Food Experience:** The accuracy score is: 0.6041666666666666

**Hotel Infrastructure:** The accuracy score is: 0.897196261682243

**Hygiene & Cleanliness:** The accuracy score is: 0.7233333333333334

**Room Equipment & Amenities:** The accuracy score is: 0.8064516129032258

**Staff & Service:** The accuracy score is: 0.625

**TV & Wi-Fi:** The accuracy score is: 0.6728395061728395

**Washroom:** The accuracy score is: 0.648854961832061

**Further Steps:**

Accuracy of L2 model can further be improved by removing the unbalancing and adding more data. With large amount of training data set, we can use neural networks. We can work more on Word Embedding and feature extractions.

As manually analyzed, there was about 90% accuracy for each tag in training data. By making the training data more accurate, we can increase the accuracy of model.