Appendix

Appendix A.

Refer to the Inter-annotator guideline agreement[1] for annotation rules as per the project scope.

Doccano

Doccano is an open source text annotation tool which is used to label, or annotate text data for:

- Text Classification
- Entity Extraction
- Sequence to Sequence translation

It has an easy to use GUI for annotating text data. First, the text data is loaded onto the software by creating a new project. Then, the entities are defined by the user. Doccano gives us the liberty to add colors and custom keyboard shortcuts to the entities, so as to assist in the manual task. Then, the text is manually labelled against the set entities. Once completed, the data can be saved in JSON format, that is JavaScript Object Notation. For each annotated project, two files are saved. One contains the original text data along with the entities and tagged words. The second file gives an insight of total counts for tags of each entity in the whole project.[1]

Entity-Aspect Selection:

500 reviews were read carefully, enlisting broad categories that were reviewed. Based on the classification of the reviews, 8 entities were observed to be significant. As majority of the reviews talked about food, staff, seat, cabin, entertainment, in-flight service, off-flight service, the same naming convention was followed to name the entity groups. For example, all the words and phrases that indicate opinions about the food served in the flight are annotated under the entity "food". Following is a wordcloud for "Food" entity. A wordcloud is a graphical representation of word frequency, such that the most common, or the most repeated word has the greatest prominence by size in the figure.

Food:

Similarly, at a step further in the hierarchy, these entities were classified based on the aspect the opinion focused on. For example, most of the reviews that talked about food, notified the reader towards the food temperature, taste etc. As a result, after comparing the number and weightage of each opinion, the Food entity was classified into 3 aspects- temperature, taste and service. This categorization was chosen as it classified all the opinions about food. Following are the word clouds for all the aspects of Food entity:

Food Service:

Food taste:

Food temperature:

The main check is to note that at all times, the words in either of the categories should not match with the words in any other category. This way, a clear and distinct boundary is maintained between the categories.

Following a similar process, wordcloud for all entities are prepared:

Seat

Seat Aspects:

Comfort

Operations

Possession

Possession aspects:

Handling

Operations

Entertainment

Entertainment Aspects:

Audio

Visual

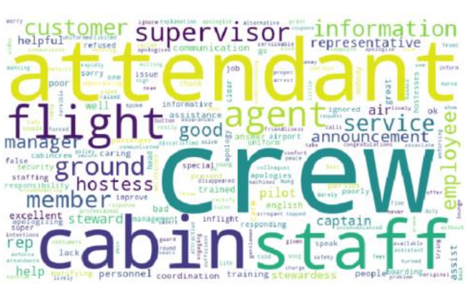
General

Staff

Staff Aspects:

Behaviour

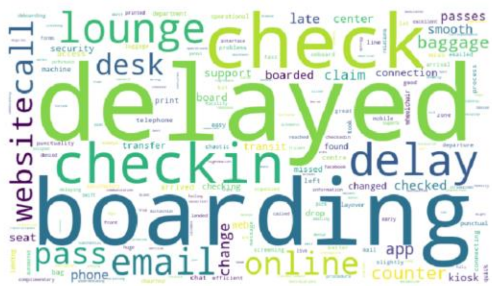
General



Off-flight Service

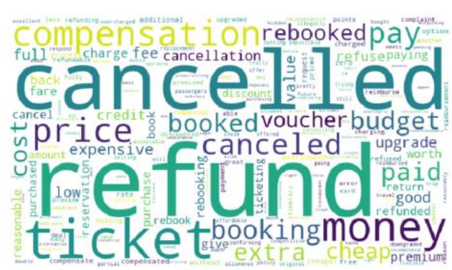
Off-flight Service Aspects:

Facility



General

Ticketing



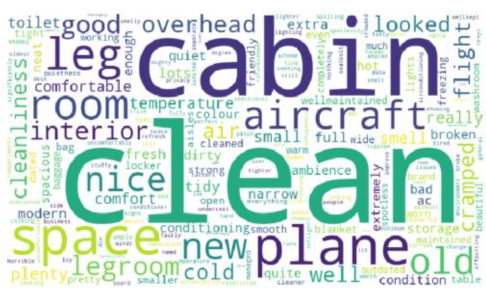
In-flight Service

Inflight Service Aspects:

Facility

Operations

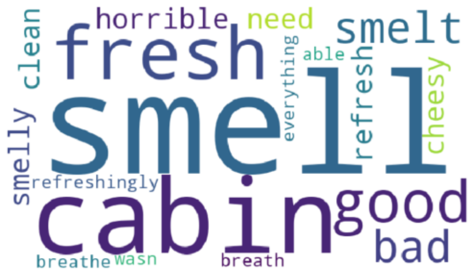
Cabin



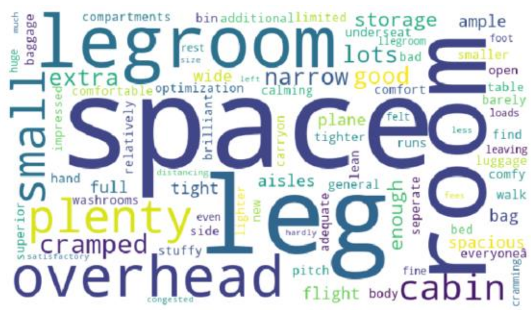
Cabin Aspects:

Condition

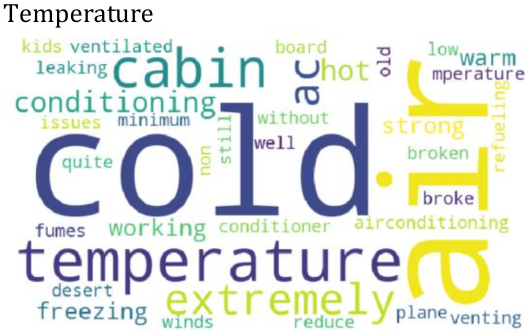
Fragrance



Size



Temperature



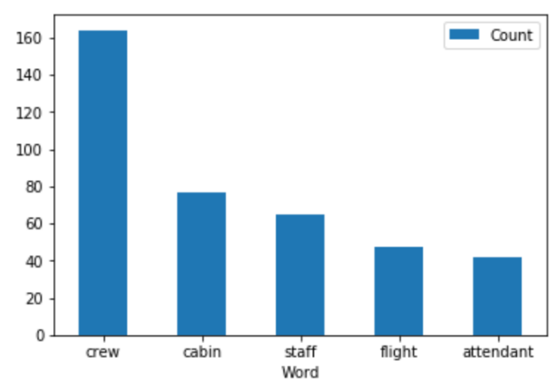


Figure: 5 most used words for “Staff” entity

Appendix B.

Cohen’s Kappa Coefficient

Since the scope of the project followed supervised learning, which requires labelled data to train the models, the process of annotating/labelling the text data with the respective entities was needed. For any supervised machine learning model to have high accuracy, it is important that the labels consistency and data integrity is maintained throughout the training data. An inter-annotator agreement was formed to make sure that both the annotators label the data in a similar manner. To keep a track of how similar the annotations are, Cohen's Kappa coefficient is used.[2]

The Kappa coefficient value for annotating entities came out to be 0.8048, whereas the Kappa coefficient value for annotating respective aspects came out to be 0.8213.

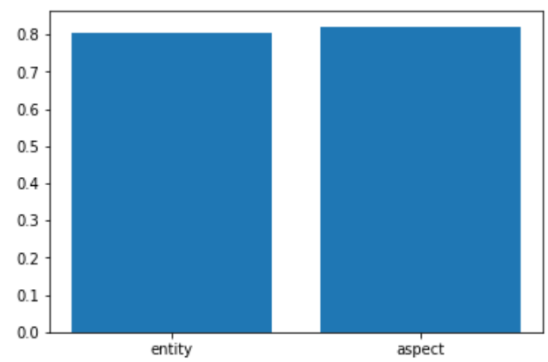


Figure: Kappa Co-efficient for entity and aspect

Furthermore, Kappa coefficient is also calculated as per each entity, to take into account any particular entity being labelled differently by the two annotators. These are plotted in a bar plot below:

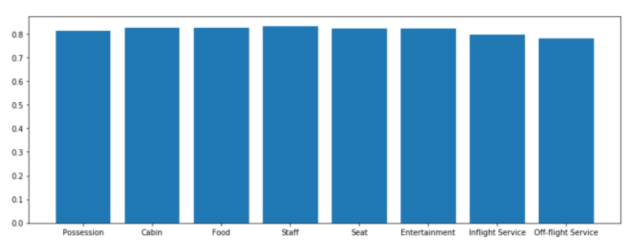


Figure: Kappa Co-efficient for all entities

Highest Kappa coefficient value is observed for 'Staff' entity, which is approximately 0.8317, and the lowest is observed for 'Off-flight Service', approximately 0.7814.

Since, all the coefficient value have a score greater than 0.75, it is safe to conclude that the annotations are quite similar in nature.

Pos Tag

The POS Tagger reads the text and assigns parts of speech to all the words appearing in the vocabulary. These parts of speech could be noun, verb, adjective etc. The Penn Treebank tag set by Stanford specifies certain naming conventions for all the parts of speech.[3]

Tanle I

Abbreviations of part-of speech tags- Penn Treebank Tag Set

Serial Number	POS Tag	Description
1	CC	Coordinating conjunction
2	CD	Cardinal number
3	DT	Determiner
4	EX	Existential *there*
5	FW	Foreign word
6	IN	Preposition or subordinating conjunction
7	JJ	Adjective
8	JJR	Adjective, comparative
9	JJS	Adjective, superlative
10	LS	List item marker
11	MD	Modal
12	NN	Noun, singular or mass
13	NNS	Noun, plural
14	NNP	Proper noun, singular
15	NNPS	Proper noun, plural
16	PDT	Predeterminer
17	POS	Possessive ending
18	PRP	Personal pronoun
19	PRP$	Possessive pronoun
20	RB	Adverb
21	RBR	Adverb, comparative
22	RBS	Adverb, superlative
23	RP	Particle
24	SYM	Symbol
25	TO	*to*
26	UH	Interjection
27	VB	Verb, base form

28	VBD	Verb, past tense
29	VBG	Verb, gerund or present participle
30	VBN	Verb, past participle
31	VBP	Verb, non-3rd person singular present
32	VBZ	Verb, 3rd person singular present
33	WDT	Wh-determiner
34	WP	Wh-pronoun
35	WP$	Possessive wh-pronoun

Dependency Parsing

Dependency Parsing helps to describe the grammatical relationships between the words of a sentence, by specifying what are called "*dependencies*". These dependencies are binary relations- a grammatical relation that holds among a governor/head and a dependent. The following is the list to map the abbreviations of the Dep tags to their respective description.[4]

Table II

Universal Dependency Relations

Serial Number	**Dependency Tag**	**Description**
1	acl	clausal modifier of noun
2	advcl	adverbial clause modifier
3	advmod	adverbial modifier
4	amod	adjectival modifier
5	appos	appositional modifier
6	aux	auxiliary
7	case	case marking
8	cc	coordinating conjunction
9	ccomp	clausal complement
10	clf	classifier
11	compound	compound
12	conj	conjunct
13	cop	copula
14	csubj	clausal subject
15	dep	unspecified dependency
16	det	determiner
17	discourse	discourse element
18	dislocated	dislocated elements
19	expl	expletive
20	fixed	fixed multiword expression
21	flat	flat multiword expression
22	goeswith	goes with
23	iobj	indirect object
24	list	list
25	mark	marker
26	nmod	nominal modifier
27	nsubj	nominal subject
28	nummod	numeric modifier
29	obj	object
30	obl	oblique nominal
31	orphan	orphan
32	parataxis	parataxis
33	punct	punctuation
34	reparandum	overridden disfluency
35	root	root
36	vocative	vocative
37	xcomp	open clausal complement

<u>Appendix C.</u>

Count Vectorizer:

Here, the collection of text reviews is converted into a matrix of token counts. The basic operation of this technique is to check each word in the document and count the number of their representations and create a matrix of these counts.

For this experiment study, since the methodology does try to keep certain punctuations and special characters, a need is felt to create own tokenizer. The results for an example sentence:

Table

Count Vectorizer using Sci-kit Learn

Sentence: *'so overall I highly recommend this airline'*						
So	Over-all	I	High-ly	Recomm-end	Th-is	airline
5	3	2	1	4	6	0

TF-IDF

It is commonly referred as TF-IDF. It can be divided as two terms namely, Term Frequency and Inverse Document Frequency.

Term Frequency (TF) can be defined as a ratio of count of the word present in a sentence to the length of the sentence.

Inverse Document Frequency (IDF) can be termed as measure of rareness of a term in the corpus. Article words like "a", "an" or "the" appear in almost every corpus, but rare words might not be present in all documents.

Appendix D.

Word Embedding

Word embedding as the name suggests is a collective name for language modelling and feature engineering techniques of Natural Language Processing. In this technique, the word phrases are mapped to vectors of real numbers.[5]

Before going in the details of our methodology for implementation of word embeddings, there are certain terminology that needs to be understood in context of word embeddings.

Language Model: The concept of a language model has a probabilistic character. It is essentially described as a function that provides a probability distribution of strings drawn from a vocabulary[1].

Vector Space Models: An algebraic model to represent text documents as vector of identifiers. Documents can be represented as

$d_j = (w_{1,j}, w_{2,j}, w_{3,j}, .., w_{t,j})$, wherein each dimension is a separate term in the document.

Distributional Semantics: In 1954, Harris stated that the basis of distributional semantics is distributional hypothesis i.e. similarity of distribution in linguistics is resulted by similarity in meaning.

n-gram: They are essentially sequencing of characters or words extracted from a text. It can be deduced as a set of n consecutive characters from a word.

Since this experiment study has limited and a small size of corpus, a decision was made for using pre-trained Twitter Glove vectors. The approach for this experiment study includes training a Word2Vec model for the experiment corpus on-top of the pre-trained Twitter Glove vectors.[6]

CBOW or Continuous Bag of Words: It is a methodology that tends to predict the probability of a word given a context. A context can either be a single or a group of words. The objective function of CBOW language model is as follows

$$J_\Theta = \frac{1}{T}\sum_{t=1}^{T} \log\ p\ (w_t |\ w_{t-n}, .., w_{t-1}, w_{t+1}, .., w_{t+n})$$

Where, a training corpus containing a sequence of T training words $w_1, w_2, w_3, \ldots, w_T$ that belongs to vocabulary V of size |V| and Θ is the parameters of the model.

Advantages of using CBOW:

1. Generally, it performs superior to deterministic methods because of its probabilistic nature.

2. Unlike a co-occurrence matrix, it does not have huge RAM requirements.

Limitations of using CBOW:

1. For example, the word Apple can mean both fruit and company. CBOW will take an average of both contexts and place it in the middle of a cluster of both these entities.

2. Optimization is highly important, else the training using a CBOW model will take forever.

Skip Gram: The aim of a skip gram language model is to predict the context given a word. It follows the inverse of CBOW's architecture. In simpler terms, skip-gram model will use the centre word

[1] Vocabulary: Set of unique words in a text corpus is referred to as a vocabulary.

to predict the surrounding words, unlike a CBOW model which uses surrounding words to predict centre word.

The skip-gram objective function sums up the log probabilities of the surrounding n words to the right and left of the target word w_t and can be represented as below

$$J_\Theta = \frac{1}{T}\sum_{t=1}^{T}\sum_{-n\leq j\leq n} \log\ p\left(w_{t+j}\middle|w_t\right)$$

So, instead of computing P (w_t) target word given w_{t+j} surrounding words, skip-gram computes surrounding word given target word.

Negative Sampling:

Let's consider, a pair (w, c) where w and c determine word and context respectively.

If the pair of word and context derive from the training data then it can be notated as

P (D=1 | w,c) - (a)

and if the word pair does not come from training data then it can be simply represented as

P (D = 0 | w,c) – (b)

So, from equations a & b, one can rewrite b as

P (D=0 | w,c) = [1 – P(D=1 | w,c)]

Assuming, there are Θ parameters controlling this distribution and can be represented as follows,

P (D = 1 | w,c, Θ)

The goal is to make all observations come from training data. And in order to do so, we have to maximise this probability and it can be denoted as below

$$\arg max_\Theta\ \Pi_{(w,c)\in D}\ P\ (D = 1\ |w,c;\Theta)$$

$$= \arg max_\Theta \log \Pi_{(w,c)\in D}\ P\ (D = 1\ |w,c;\Theta)$$

$$= \arg max_\Theta \sum_{(w,c)\in D} P(D = 1\ |w,c;\Theta)$$

Using soft-max[2] distribution, above equation can be rewritten as follows,

$$\mathrm{P}\ (\mathrm{D}\ =\ 1\ |\mathrm{w,c;}\ \Theta) = \frac{1}{1+\ e^{-Vc*Vw}}$$

This can be represented as objective function as follows,

$$\arg max_\Theta = \sum_{(w,c)\in D} \log\frac{1}{1+\ e^{-Vc*Vw}}$$

The only limitation of the above is that it allows same (w,c) pair combinations to occur.

So, ahead a mechanism will be developed that prevents vectors with same value. This can be achieved by introducing (w,c) pairs that are not in the data. So generate new pairs which are not in training data and are represented as below

$$D^1 = random(w,c)pairs$$

Since, these pairs are assumed to be incorrect, this approach is named as negative sampling and the objective function can now be optimized as below,

$$\arg max_\Theta\, \Pi_{(w,c)\in D}\ p(= 1\ |c,w;\ \Theta)\ .\Pi_{(w,c)\in D}\ P(D = 0\ |c,w;\ \Theta)$$

$$= \arg max_\Theta\, \Pi_{(w,c)\in D}\ p(D = 1\ |c,w;\ \Theta)\ .\,\Pi_{(w,c)\in D}\ [1 - P(D = 1\ |\ c,w;\ \Theta)$$

$$= \arg max_\Theta \sum_{(c,w)\in D^1} \log P\ (D = 1|c,w;\Theta) + \sum_{(c,w)\in D^1} \log P\ (D = 0|c,w;\Theta)$$

$$= \arg max_\Theta \sum_{(c,w)\in D^1} \log P\ (D = 1|c,w;\Theta) + \sum_{(c,w)\in D^1} \log[1 -\ P\ (D = 0|c,w;\Theta)\ ]$$

[2] Soft-max: It is a normalized exponential function that takes vector a of R real numbers as input and normalizes it into a probability distribution consisting of R probabilities proportional to the exponential of input numbers

$$= \arg max_{\Theta} \sum_{(c,w)\in D^1} \log \frac{1}{1+ e^{-Vc*Vw}} + \sum_{(c,w)\in D^1} \log[1 - \frac{1}{1+ e^{-Vc*Vw}}]$$

$$= \arg max_{\Theta} \sum_{(c,w)\in D^1} \log \frac{1}{1+ e^{-Vc*Vw}} + \sum_{(c,w)\in D^1} \log \frac{1}{1+ e^{Vc*Vw}}$$

Replacing, $\frac{1}{1+ e^{-x}}$ *by* $\sigma(x)$, *we get*

$$\arg max_{\Theta} \sum_{(c,w)\in D^1} \log \frac{1}{1+ e^{-Vc*Vw}} + \sum_{(c,w)\in D^1} \log \frac{1}{1+ e^{Vc*Vw}}$$

$$= \arg max_{\Theta} \sum_{(c,w)\in D^1} \log\ \sigma(V_c . V_w) + \sum_{(c,w)\in D^1} \log\ \sigma(-V_c . V_w)$$

The aim is to represent that $D \cup D^1$ depicts the entire corpus.

Context Window

The context window determines which contextual neighbors are taken into account when estimating the vector representations

context window is the maximum window size (i.e. the maximum distance between the focus word and its contextual neighbors). This parameter is the easiest one to adjust using existing software, which is why it is comparatively well studied. Larger windows are known to induce embeddings that are more 'topical' or 'associative', improving their performance on analogy test sets, while smaller windows induce more 'functional' and 'synonymic' models, leading to better performance on similarity test sets.

Visualizing elements of the Word2Vec Model

Cosine Similarity- computes similarity between a simple mean of the projection weight vectors of the given words and the vectors for each word in the model. The method corresponds to the word-analogy and distance scripts in the original word2vec implementation. It is a metric used to measure how similar the documents are irrespective of their size

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\ \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2}\ \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

The higher the value of COS Theta, the higher the similarity.

```
def cosine_distance_wordembedding_method(s1, s2):
    import scipy
    vector_1 = np.mean([w2v[word] for word in s1],axis=0)
    vector_2 = np.mean([w2v[word] for word in s2],axis=0)
    cosine = scipy.spatial.distance.cosine(vector_1, vector_2)
    print('Word Embedding method with a cosine distance asses that our two sentences are similar to',round((1-cosine)*100,2),'%')
```

Figure: Code to calculate Cosine Similarity

The result for the Adjective-Noun pairs comes out to be 73.21%.

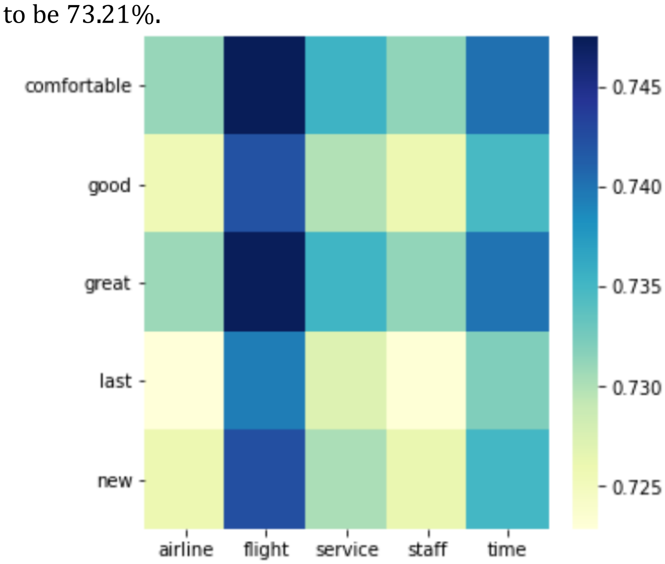


Figure: Heatmap between Adjectives and Nouns

T-SNE (t-distributed stochastic neighboring embedding)

T-SNE is quite useful in case it is necessary to visualize similarity between objects which are located into multidimensional space. With a large dataset, it is becoming more and more difficult to make an easy-to-read t-SNE plot, so it is common practice to visualize groups of the most similar words.

Hyperparameters of T-SNE:

- **perplexity**: It is a value which in context of T-SNE, may be viewed as a smooth measure of the effective number of neighbours. It is related to the number of nearest neighbours that are employed in many other manifold learners
- **n_components**: dimension of the output space
- **n_iter**: Maximum number of iterations for optimization
- **init**: Initialization of embedding matrix

The following visualization can be useful to understand how Word2Vec works and how to interpret relations between vectors captured from your texts before using them in neural networks or other machine learning algorithms:

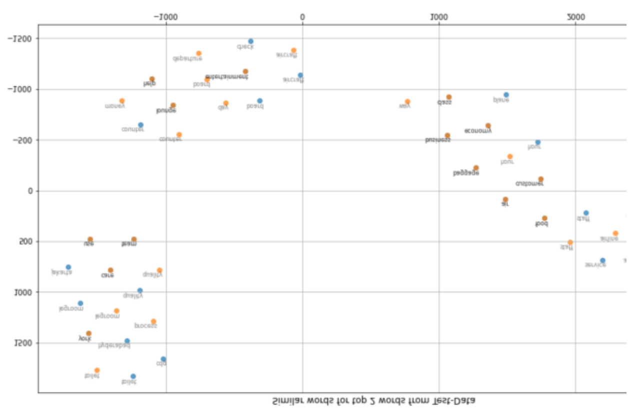


Figure: Words in vector space

Interpretation:

From the Test Dataset, using TF-IDF we found that the words "Food" and "Hour" are most common. So, to find the words in the embedding that are most associated with these two words, we plotted a TSNE-plot. As, described before, TSNE finds the nearest neighbor embedding for the words and thus, the TSNE plotted shows clusters of words that are closely embedded together. Orange highlights the words that are associated for the word-HOUR, Blue highlights the words that are associated for the word-FOOD. and the Brown highlighted words are associated with both the words Hour and Food

Appendix E.

Conditional Random Fields:

In sequential labelling or learning, previously most of the work was done using two machine learning approaches. One of which was a generative probabilistic method and the other was a sequential classification method.

The generative probabilistic method depends on k-order generative probabilistic models of paired input and label sequences using either Hidden Markov Models or Multi-level Markov Models. This approach though provides a good training and decoding algorithms of Markov Models it requires more strict conditional independence assumptions. Thus, making it impractical to use a windowed sequence of input as well as surrounding labels to make a label dependent on such a sequence.

As demonstrated in work of maximum-entropy by McCallum and Ratnaparkhi, many correlated features can be handled by a sequential classifiers like linear-classifiers, AdaBoost and support vector machines. Generative models can trade off decisions at different positions against one another, this cannot be done by Sequence Classifiers. This compelled even the best sequential learning classifiers to use heuristic combinations of forward-moving and backward-moving sequential classifiers.

Conditional Random fields brings the best out of both worlds of generative probabilistic modelling and sequential label classification.[7]

It can adjust to a variety of statistically correlated features as input just like a sequential label classifier. And just like a generative probabilistic model it can trade off decisions at different sequence to obtain a global optimal labelling.

Lafferty et al. defined conditional random field on a set of X observations with a set of Y labels, for example X might range over sentences and Y might range over part-of-speech tags. These random variables X and Y are jointly distributed, but in a discriminative framework, a conditional model is constructed p(Y | X) from paired observations and label sequences.

The principle is based on the fact that the conditional probability of a label Y_v depends on a label Y_w if and only if there is affinity with Y_v

The joint distribution over the label sequences Y given X has the form:

$P\,\Theta\,(y \mid x) \propto \exp\,(\sum_{e \in E,K} \lambda_k\, f_k\,(y \mid e,x) + \sum_{v \in V,K} \mu_k\, g_k(v, y|v, x)$ – (2)

where x is data sequence,

y is label sequence,

y |s is the set of components of y associated with vertices in subgraph S, f_k and g_k are feature functions and Θ is the set of weight parameters.

$\Theta = (\lambda_1, \lambda_2, \lambda_3,\ldots;\ \mu_1, \mu_2, \mu_3,..)$

Typically to the subset of {0,1}, the feature functions f_k and g_k maps a set of observations X to a real number. The feature functions are built in such a way that the observations X_i are modelled as a vector. These are usually hand-crafted Boolean values.

Appendix F.

Classification algorithms

SVM

For defining the hyperplane, the following equation is used,

$$w^T.x + b = 0$$

where, w denotes weight vector, x is the input vector and b is bias.

This helps in creating a hyperplane with as big a margin as possible.[8]

Decision Tree

In the beginning of this algorithm, the whole training dataset is the root of the tree, where root node represents the entire population. Each box represented in the above figure is a node at which tests (T) are applied to recursively split the dataset in smaller groups. The letters (A, B, C) at each leaf node represent the labels assigned to every observation.

The test (T) is basically making the best choice to reduce the entropy to minimum and thereby improving information gain to maximum. This process is carried recursively till entropy is minimized among all branches of the tree.[9]

Entropy and information gain are calculated as follows,

$$Entropy = \sum_{i=1}^{c} -p_i.\,log_2 p_i$$

$$Information\ Gain = Entropy_{before-split} - Entropy_{after-split}$$

Boosting

It is an implementation of gradient boosted decision trees.[10]

For a given dataset, with n examples and m features $D = \{(x_i, y_i)\}, (|\mathrm{D}| = \mathrm{n}, \mathrm{xi} \in \mathrm{R\ m}, \mathrm{yi} \in \mathrm{R})$, the output predicted by such a tree ensemble technique can be depicted as below,

$$y^T{}_i = \varphi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in F$$

where $F = \{f(x) = w_{q(x)}\}\{q: R^m \rightarrow T, w \in R^T\}$ describes the space of the trees.

Random Forest

Random Forest is essentially an ensemble classifier that uses several decision trees and then outputs the class that is predicted by the maximum number of trees. It is a robust method andproves to output high accuracy, because of it not being dependent on any particular decision tree, but abunch, or forest of them. The idea implements Breiman's "bagging" technique , which is a way to decrease the variance of the prediction by generating supplementary data o train from dataset using several combinations with repetition, therefore producing multi-sets of the original data.[11]

Voting Classifier

Voting Classifier is an ensemble technique which is based on a simple working mechanism, that is 'voting'. Several different algorithms are trained on the dataset, and the output of each is combined to predict the final class. It works on a 'majority' principle, and the class being predicted by the greatest number of classifiers, is chosen as the ensemble result for the data. The models used were decision trees, random forest and extra trees classifier. Extra trees classifier, or extremely randomized trees uses all the data available in the training set to build each decision tree with depth set to one, also called as stump. Furthermore, the best split to form the root node or any other node is determined by searching in a subset of randomly selected features having size equal to square root of the number of features. For each selected feature, the split is chosen randomly. Therefore, the degree of randomness is more extreme than that of random forest. Thus, although decision tree, random forest and extra trees, all

implement decision trees, they have different understanding of the data. Hence, the output of each of these classifiers is taken into consideration and the class predicted the maximum number of times is voted as the final predicted class.[12]

ADASYN

It was observed that some aspects inspite of being important, were not talked about much. For example, food temperature is an important aspect of food, but the reviews containing food temperature aspect were quite less in number than that of the reviews talking about food taste. Similarly, reviews containing cabin fragrance aspect were less in number than the reviews containing cabin condition aspect. Such a difference in numbers would create an unwanted bias in the model, increasing the chances of overfitting. To overcome this problem, a technique known as ADASYN was used. Adaptive Synthetic Sampling Method for Imbalanced Data is an oversampling technique for minority classes which tackles the imbalanced classification problems. In ADASYN, the degree of imbalance, d is calculated by dividing the number of minority class samples by the number of majority class samples.

Consider Inflight-Service, which has two aspects-Operations and Facility. The initial count of data for

these two aspects was 327 and 263 respectively, which clearly indicates imbalance. The degree of

imbalance d would be 263/327 which is 0.8 . For each datapoint in minority dataset, synthetic data is

generated so as to bring the count closer to the majority class data count, while avoiding high

redundancy. The algorithm increased the minority class datapoint count to 278, thereby increasing the

degree of imbalance to 278/327 which is 0.85, bring the number of datapoints for 'facility' closer to

'operations'.

Appexdix G.

The steps taken for pre-processing the text are described in section.

Given a sequence of words, we construct vector of features for each of the labelled words. These vectors describing the features contain the following encoded features

1. Lower – case of word
2. previous word
3. next word
4. Word Length
5. Part-of-speech tag
6. Dependent Word
7. Dependent Word Part-of-speech tag
8. Dependency Tag

It is crucial to understand the fact that the stopwords removal step is both, a boon and a bane, as removal of these words leads to breakage of the sentence structure, making it difficult to analyze the text semantically. Therefore, in dependency parsing step, the text was used without removing the stopwords. Another part of preprocessing text is dealing with contraction, which means shortening of words or syllables. It was noticed that several words were present in the data in many different forms, for instance, the term “could not” was present in terms of “couldn’t” as well. These contractions occur depending upon the tone of the reviewer or the context of the review. It is often seen that the implied meaning of the phrase does not differ, but the model considers them as different words, leading to poor training. Therefore, the need arises to alter the text in such a way that the model links up the different variations that have the same implied meaning. In this example, we change the term “couldn’t” to “could not”. Such expansion of contracted terms helps with text standardization. Apart from this, all the text is changed to lowercase, to create a uniform text dataset, which initially contained a mixture of uppercase and lowercase texts. Additionally, numerals are converted to words, for example- ‘$3000’ is changed to ‘three thousand dollars’.

Corpus can be defined as a collection of textual data, or a body of writing, that is based around a particular subject. The reviews after the above steps are added collectively to a list of reviews, henceforth referred to as “Corpus”. This corpus could be thought of a collection of all the scraped data, for all the airlines, referring to many different entities and opinions- after cleaning and preprocessing. This corpus serves as a basis of document for further steps.

Appendix H.

TTR

There are some rules for calculating TTR, which are adapted in this study. These rules include following,

a) Compound nouns and hyphen words are considered as one word

b) Parts of verbal phrases are considered as separate words, example, phrase like “*meals were served”* counts as three tokens, *meals, were and served*

c) Contractions are considered as two words, example *couldn’t,* is counted as *could not*

Results of TTR

Since, the present study is for user generated data for airlines, it is expected that there will be words that might be repeated quite often. Data is gathered for 16 airlines from two different websites and the type token ratio is observed to be between 0.2 to 0.6 for almost all airlines.

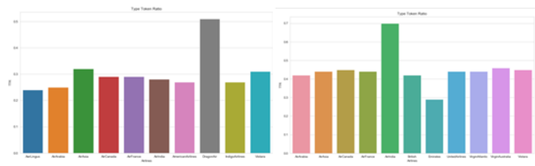


Figure: 5 most u

Type token ratio between both data sources is observed to be 0.27, which means that there are many words that are repeated between them.

Zipf’s other law states that the number of meanings (m) of a word is the square root of its frequency.

$$Given\ first\ law, m \propto \frac{1}{\sqrt{m}},$$

$$m \propto \sqrt{\mathrm{f}}$$

This means that the second most repeated word will have a frequency that is half of the first word and the third most repeated word will have a frequency that is half of the second most repeated word.

As seen below, our corpus does follow Zipf’s distribution.[13]

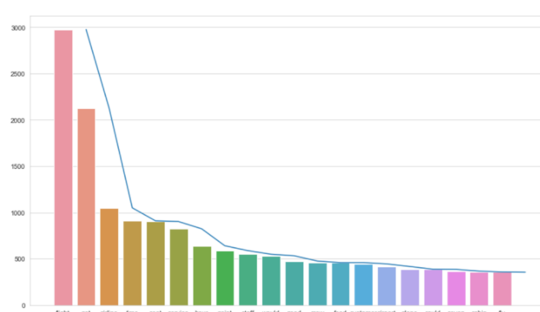


Figure: 5 most u

Appendix I.

The project can majorly be divided into these parts- Entity extraction, Aspect identification/extraction, sentiment analysis. Several parameters are used to check the level of

righteousness of the project.

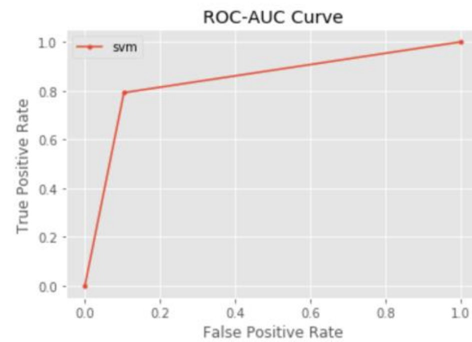
A point to be pondered about is as to which of the performance metrics should be taken into account, to better judge the model. The most common idea, "accuracy" works best when the false positives and false negatives have similar cost. However, the airline reviews contained an unequal number of positive and negative opinions for different aspects- because opinions are a subjective matter and could differ for any two people. Therefore, the performance metrics used were F1, precision and recall. These are defined below:

Precision: The measure of the correctly identified positive cases from collectively all the predicted positive cases. It is beneficial when the costs of False Positives is high.

Recall: The measure of the correctly identified positive cases from collectively all the actual positive cases. It is significant when the cost of False Negatives is high. Mutually, F1 score is the weighted average of Precision and Recall, and takes both false positives and false negatives into account. Therefore, it proved to be the best choice.

Performance metrics for "Food" entity based on different approaches, simultaneously applying SMOTE:

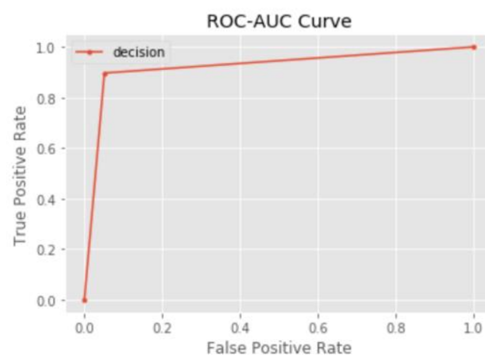
1. SVM



Classification Report is as follows

	precision	recall	f1-score	support
food_service	0.86	0.65	0.74	310
food_taste	0.75	0.88	0.81	317
food_temperature	0.78	0.89	0.83	152
accuracy			0.79	779
macro avg	0.80	0.81	0.80	779
weighted avg	0.80	0.79	0.79	779

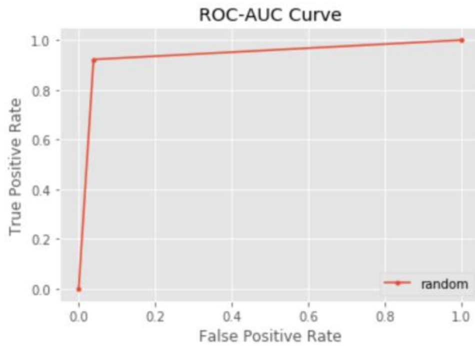
2. Decision Tree



Classification Report is as follows

	precision	recall	f1-score	support
food_service	0.89	0.87	0.88	318
food_taste	0.87	0.90	0.88	300
food_temperature	0.98	0.95	0.96	139
accuracy			0.90	757
macro avg	0.91	0.91	0.91	757
weighted avg	0.90	0.90	0.90	757

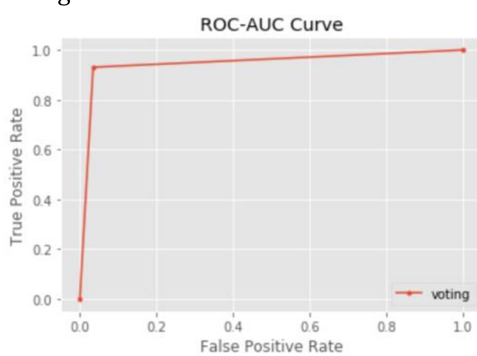
3. Random Forest



Classification Report is as follows

	precision	recall	f1-score	supp
food_service	0.91	0.92	0.91	
food_taste	0.91	0.92	0.91	
food_temperature	0.99	0.94	0.96	
accuracy			0.92	
macro avg	0.93	0.92	0.93	
weighted avg	0.92	0.92	0.92	

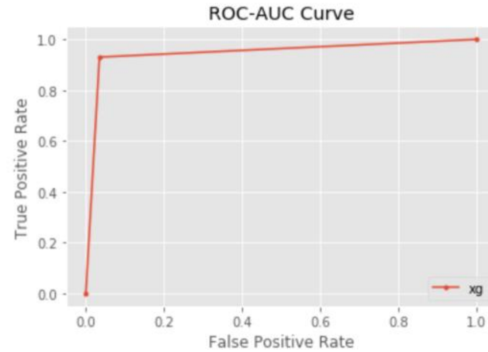
4. Voting Classifier



Classification Report is as follows

	precision	recall	f1-score	supp
food_service	0.91	0.91	0.91	
food_taste	0.91	0.93	0.92	
food_temperature	0.99	0.97	0.98	
accuracy			0.93	
macro avg	0.94	0.93	0.94	
weighted avg	0.93	0.93	0.93	

5. XG-Boost



Classification Report is as follows

	precision	recall	f1-score	support
food_service	0.94	0.90	0.92	318
food_taste	0.91	0.95	0.93	305
food_temperature	0.97	0.94	0.96	108
accuracy			0.93	731
macro avg	0.94	0.93	0.94	731
weighted avg	0.93	0.93	0.93	731

References

[1] *doccano/doccano*. doccano, 2020.

[2] A. Rosenberg and E. Binkowski, “Augmenting the kappa statistic to determine interannotator reliability for multiply labeled data points,” in *Proceedings of HLT-NAACL 2004: Short Papers*, Boston, Massachusetts, USA, May 2004, pp. 77–80, Accessed: Aug. 15, 2020. [Online]. Available: https://www.aclweb.org/anthology/N04-4020.

[3] Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 63-70.

[4] Moore R, Caines A, Graham C and Buttery P Incremental Dependency Parsing and Disfluency Detection in Spoken Learner English Proceedings of the 18th International Conference on Text,

Speech, and Dialogue - Volume 9302, (470-479)

[5] Y. Goldberg and O. Levy, “word2vec Explained: deriving Mikolov et al.’s negative-sampling word-embedding method,” *arXiv:1402.3722 [cs, stat]*, Feb. 2014, Accessed: Apr. 22, 2020. [Online]. Available: http://arxiv.org/abs/1402.3722.

[6] J. Pennington, R. Socher, and C. Manning, “GloVe: Global Vectors for Word Representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1532–1543, doi: 10.3115/v1/D14-1162.

[7] J. Lafferty, A. McCallum, and F. C. N. Pereira, “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data,” p. 10.

[8] J. A. K. Suykens and J. Vandewalle, “Least Squares Support Vector Machine Classifiers,” *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, Jun. 1999, doi: 10.1023/A:1018628609742.

[9] S. R. Safavian and D. Landgrebe, “A survey of decision tree classifier methodology,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 660–674, May 1991, doi: 10.1109/21.97458.

[10] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco California USA, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[11] A. Cutler, D. R. Cutler, and J. R. Stevens, “Random Forests,” in *Ensemble Machine Learning: Methods and Applications*, C. Zhang and Y. Ma, Eds. Boston, MA: Springer US, 2012, pp. 157–175.

[12] S. Saha and A. Ekbal, “Combining multiple classifiers using vote based classifier ensemble technique for named entity recognition,” *Data & Knowledge Engineering*, vol. 85, pp. 15–39, May 2013, doi: 10.1016/j.datak.2012.06.003

[13] D. M. W. Powers, “Applications and Explanations of Zipf’s Law,” 1998, Accessed: Aug. 16, 2020. [Online]. Available: https://www.aclweb.org/anthology/W98-1218.