Appendix
Implicit aspect-based opinion mining and analysis of airline industry based on user generated reviews

Kanishk Verma
School of Computing
Dublin City University
Dublin, Ireland
kanishk.verma@dcu.ie
0000-0001-7172-4098

Appendix

1 List of Tables

2 List of Equations

3 List of Algorithms

4 List of Figures

A. Appendix A

1. **List of Airlines**

Sr no.	Airline	Website(s)
1	Aer Lingus	Airline Ratings
2	Air Arabia	Airline Ratings, Trip Advisor
3	Air Asia	Airline Ratings, Trip Advisor
4	Air France	Airline Ratings, Trip Advisor
5	Air Canada	Airline Ratings, Trip Advisor
6	Air France	Airline Ratings, Trip Advisor
7	Air India	Airline Ratings, Trip Advisor
8	Vistara	Airline Ratings, Trip Advisor
9	British Airways	Trip Advisor
10	Emirates	Trip Advisor

11	American Airlines	Airline Ratings
12	United Airlines	Trip Advisor
13	Virgin Atlantic	Trip Advisor
14	Virgin Australia	Trip Advisor
15	Indigo Airlines	Airline Ratings
16	Dragon Air	Airline Ratings

1.1. Website-wise Airlines

Website	Number of Airlines
Trip Advisor	11
Airline Ratings	10
Common	6

2. Annotation Guidelines

Following a supervised learning approach for the scope of this study, labelled data was required to train the machine learning and ensemble learning models to identify, extract and classify airline-specific implicit aspects and opinions. Since, the data was fresh, new and one of a kind, it was important to manually annotate and label the data.

With two annotators, manually annotating about 1803 reviews, there was a need to metricize the agreement level between the annotators which would act as the basis of ground truth for this project scope.

Refer to the inter-annotator guideline agreement document for detailed annotation rules for the scope of this research study. (See Inter-annotator agreement)

3. Kappa Co-efficient

Cohen's Kappa co-efficient is a method to measure inter-rater reliability for categorical data. (Rosenberg and Binkowski, 2004). The way static kappa co-efficient is calculated is as follows,

$$K = \frac{p(A) - p(E)}{1 - p(E)}$$

Where, K is the kappa value, p(A) is the probability of the actual outcome and p(E) is the probability of the expected outcome. (Rosenberg and Binkowski, 2004)

For this study, the method to calculate kappa value was calculated is the same described in the paper by *Rosenberg and Binkowski et. al.* section 5. (Rosenberg and Binkowski, 2004)

The results for this study for two annotators range from 0.8048 and 0.8213. Since, the annotation was performed on two levels, the kappa value of 0.8048 indicates the inter-annotator agreement level for Entity-wise labelling. The kappa value of 0.8213 indicates the inter-annotator agreement level for implicit-aspect wise labelling. Figure 1. gives a graphical representation of the same.

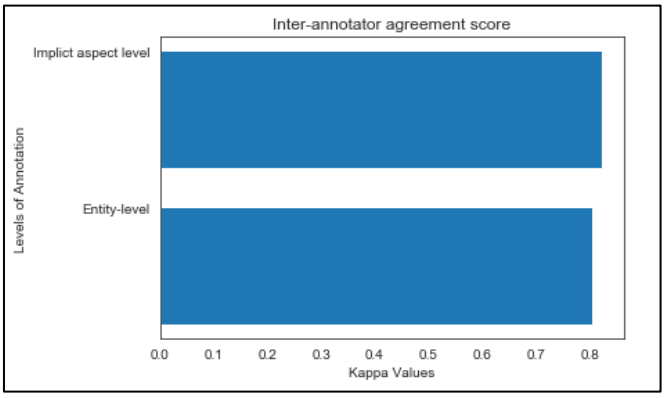


Figure 1. Kappa level-wise score

Detailed kappa values for each entity-wise labelling is available in figure 2.

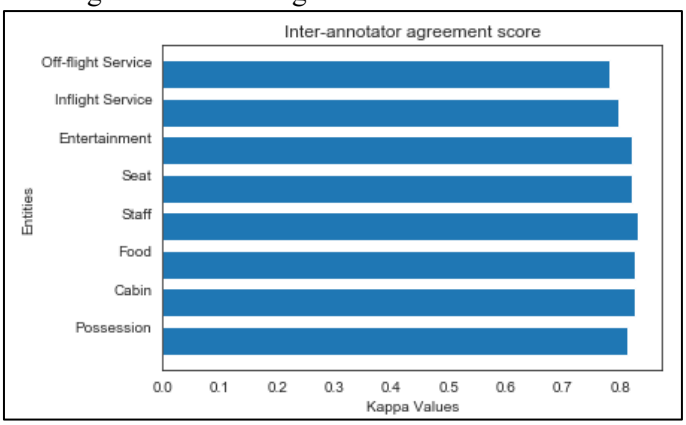


Figure.2. Kappa entity-wise score

B. Appendix B

1. Feature Engineering Tasks

Detailed description of all feature engineering tasks is as below,

1.1. Word Features
  1.1.1. Part-Of-Speech Tags
  1.1.2. Dependency Parsing
1.2. Numeric Features
  1.2.1. Count Vectorizer
  1.2.2. Term frequency – inverse document frequency
  1.2.3. Augmenting word embeddings

Part-of-speech Tags

Parts-of-speech or POS indicates how a word in a sentence functions both grammatically and contextually i.e. what it means. (Ratnaparkhi, 1996)

In natural language processing Stanford university devised and developed a part-of-speech tagging methodology. When a list of sentences or paragraphs are parsed through the product of this methodology, it automatically assigns a part-of-speech tag to each word in the sentence. ("Document (Stanford CoreNLP API)," n.d.)

In table 1, one can find a list of part-of-speech tags that are assigned to each word and what it indicates. This has been adapted from the Penn Treebank. ("Penn Treebank P.O.S. Tags," n.d.)

Tag	Description
CC	Co-ordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
NN	Noun
NNS	Noun, plural
NNP	Proper Noun, singular
NNPS	Proper noun, plural
PRP	Personal noun
VB	Verb base form
VBD	Verb past tense

Table 1. POS tag list

Example for this study,
Input: *"Overall the experience was comfortable and spacious with delicious meals"*

POS-Tags: *[('overall', 'JJ'), ('experience', 'NN'), ('comfortable', 'JJ'), ('spacious', 'JJ'), ('delicious', 'JJ'), ('meals', 'NNS')]*

Here the tags "JJ", "NN" and "NNS" mean adjective, noun and singular noun respectively.

Dependency Parsing

Adopted from the early works of French Linguist Lucien Tesnière, dependency grammar is a notion that words are connected to each other by directed links. Dependency parsing is a technique which extracts such a dependent relation between words in a sentence(s) or paragraph(s). Universal dependencies developed a framework allows one to parse raw text and get these dependent relations between words in the text. This framework is available through Stanford's CORE NLP API [1]. ("Document (Stanford CoreNLP API)," n.d.)

For this study, the result of the following sentence can be found below,

Input: *"Overall the experience was comfortable and spacious with delicious meals"*

Output: *[('overall', 'advmod'), ('comfortable', 'ROOT'), ('the', 'det'), ('experience', 'nsubj'), ('was', 'cop'), ('and', 'cc'), ('spacious', 'conj'), ('with', 'case'), ('meals', 'obl'), ('delicious', 'amod')]*

This can be visualized and understood by a dependency graph tree using GraphViz as seen in figure 3.

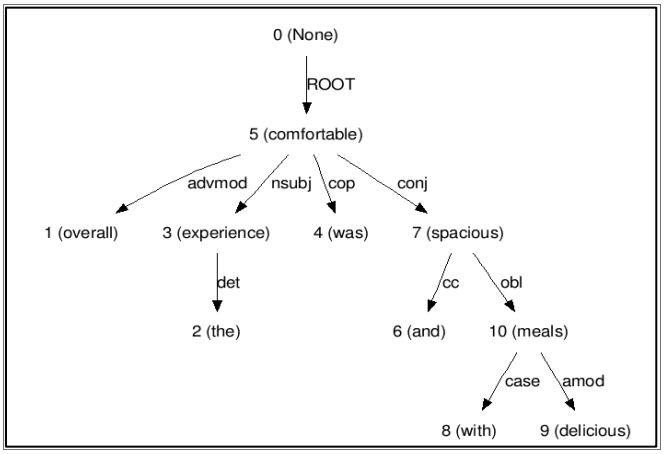


Figure 3. Dependency tree graph

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,

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

[1] API: application programming interface.

Table 2. Count Vectorizer example

Term frequency – inverse document frequency
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.

Augmenting Word Embeddings
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.

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[2].
Vector Space Models: An algebraic model to represent text documents as vector of identifiers. Documents can be represented as (Bengio et al., 2001)

$$d_j = (w_{1,j}, w_{2,j}, w_{3,j}, .., w_{t,j})$$

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.(Harris, 1954)

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.(Majumder et al., n.d.)
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.(Pennington et al., 2014)

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(Paltoglou and Thelwall, 2013)

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

Where, a training corpus containing a sequence of T training words w_1, w_2, w_3, …, 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 center word to predict the surrounding words, unlike a

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

CBOW model which uses surrounding words to predict center word.(Barazza, 2017)

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

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

So, instead of computing P (w_t) target word given w_{t+j} surrounding words, skip-gram computes surrounding word given target word.(Barazza, 2017)

Negative Sampling: In the year 2013 Mikolov et al. (Mikolov et al., 2013) deduced an efficient method to derive meaningful word embeddings using negative sampling. Though based on Skip-gram model, it is optimizing a different objective.

Let's consider, a pair (w, c) where w and c determine word and context respectively.(Mikolov et al., 2013)

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

$$\mathrm{P}\,(\mathrm{D} = 1 \mid \mathrm{w}, \mathrm{c}) \quad [\mathrm{a}]$$

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

$$\mathrm{P}\,(\mathrm{D} = 0 \mid \mathrm{w}, \mathrm{c}) \quad [\mathrm{b}]$$

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

$$\mathrm{P}\,(\mathrm{D} = 0 \mid \mathrm{w}, \mathrm{c}) = [\,1 - \mathrm{P}(\mathrm{D} = 1 \mid \mathrm{w}, \mathrm{c})\,]$$

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

$$P\,(D = 1 \mid w, c, \Theta\,)$$

The goal is to make all observations come from training data. And in order to do so, we must maximize this probability and it can be denoted as below(Goldberg and Levy, 2014)

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

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

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

Using soft-max[3] distribution, above equation can be rewritten as follows,(Goldberg and Levy, 2014)

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

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(Goldberg and Levy, 2014)

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

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,

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

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

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

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

[3] 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.

In the work of Mikolov et al. (Mikolov et al., 2013) each context is a word and all words appear as context and so $P_{context}(x) = P_{words}(x) = \frac{count(x)}{|text|}$ (Goldberg and Levy, 2014)

Context Window: The context window thus determines which contextual neighbours are accounted when estimating the vector

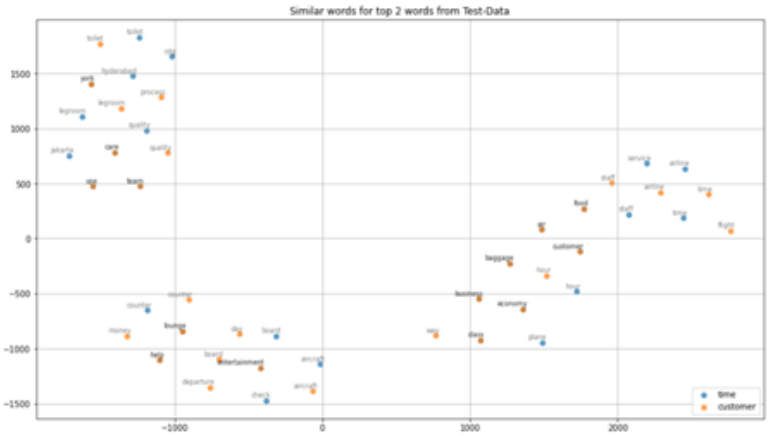


Fig 4. TSNE

representations context window is the maximum window size (i.e. the maximum distance between the focus word and its contextual neighbours). 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 (Goldberg and Levy, 2014)

C. Appendix C

D. Appendix D

1. TSNE

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: (Smetanin, 2018)

- **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 (Smetanin, 2018)
- **n_components:** dimension of the output space
- **n_iter:** Maximum number of iterations for optimization
- **init:** Initialization of embedding matrix

The visualization in figure 4, 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.(Smetanin, 2018)

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 neighbour embedding for the words and thus, the TSNE plotted shows clusters of words that are closely embeded 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

Cosine Similarity Index
It 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.

E. Appendix E

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. (McCallum et al., n.d.)

As demonstrated in work of maximum-entropy by McCallum(McCallum et al., n.d.) and Ratnaparkhi (Ratnaparkhi, 1996), 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. (Lafferty et al., n.d.)
Conditional Random fields brings the best out of both worlds of generative probabilistic modelling and sequential label classification.

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.(Lafferty et al., n.d.)

The principle is because the conditional probability of a label Y_y depends on a label Y_w if and only if there is affinity with Y_y

The joint distribution over the label sequences Y given X has the form: (Lafferty et al., n.d.)

$$P\,\Theta\,(y \mid x) \;\propto\; \exp\,(\sum_{e \in E,K} \lambda k\, fk\,(y \mid e, x) + \sum_{v \in V,K} \mu k\, gk(v, y \mid v, x)$$

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.(Lafferty et al., n.d.)

Details of Hyperparameter optimized CRF

Top likely transitions:

p	-> p	4.504740
f	-> f	4.498736
e	-> e	4.313616
i	-> i	4.186723
o	-> o	4.164468
st	-> st	4.052589
c	-> c	4.040683
s	-> s	3.888215
p	-> e	-2.751032
e	-> p	-2.751032
c	-> e	-2.881683
e	-> c	-2.881683
p	-> c	-2.941894
c	-> p	-2.941894
p	-> i	-3.046743
i	-> p	-3.046743
c	-> i	-3.185999
i	-> c	-3.185999
e	-> i	-3.226524
i	-> e	-3.226524

Top unlikely transitions:

o	-> c	-4.005861
c	-> o	-4.005861
f	-> s	-4.139228
s	-> f	-4.139228
st	-> i	-4.162967
i	-> st	-4.162967
p	-> o	-4.219411
o	-> p	-4.219411
f	-> st	-4.254724
st	-> f	-4.254724
f	-> o	-4.305534
o	-> f	-4.305534
o	-> i	-4.442339
i	-> o	-4.442339
s	-> st	-4.664813
st	-> s	-4.664813
s	-> o	-4.675678
o	-> s	-4.675678
o	-> st	-4.840923
st	-> o	-4.840923

Top positive:

1.283725 c	previousWord:underseat
1.204137 c	previousWord:everyone's
0.985568 i	previousWord:pillow
0.883634 c	nextWord:comfort
0.873309 f	depWord:awesome
0.857807 p	depWord:infuriate
0.853878 i	depWord:perth
0.846330 i	depWord:face
0.819825 f	depWord:wine
0.799197 i	previousWord:slot
0.799132 p	depWord:sigh
0.779110 p	depWord:view
0.776045 s	depWord:stuck
0.774272 s	previousWord:thirtysix
0.773041 p	depWord:retrieve
0.770658 s	previousWord:access
0.767794 st	nextWord:assistant
0.767100 f	depWord:glass
0.765888 e	depWord:collection
0.758168 st	depWord:rep
0.752351 i	depWord:flushing
0.749732 i	depWord:towel
0.745956 i	previousWord:boarded
0.742228 s	nextWord:awful
0.740946 f	depWord:beverages
0.739252 s	depWord:reallocation
0.738875 o	depWord:multilingual
0.729930 st	depWord:competent
0.726705 f	previousWord:try
0.720303 p	nextWord:12

Top negative:

-0.212331 p	depWord:started
-0.213428 p	depWord:went
-0.216759 p	depWordPos:IN
-0.216992 o	depWord:looked
-0.219010 f	depTag:acl:to
-0.219329 st	depWord:able

-0.220520 s previousWord:pretty
-0.223933 f previousWord:they
-0.226742 s nextWord:provided
-0.227997 f previousWord:any
-0.228220 o depWord:help
-0.235643 o depWord:bags
-0.236823 i depWord:to
-0.242775 e depWord:seat
-0.243770 o previousWord:some
-0.249494 st depWord:checkin
-0.256180 p depTag:obl:to
-0.258356 i previousWord:for
-0.259542 o previousWord:other
-0.264707 o previousWord:smooth
-0.270876 f previousWord:service
-0.273975 f depTag:compound:prt
-0.283252 i depTag:nmod:for
-0.283458 st previousWord:website
-0.294158 f previousWord:could
-0.307149 s depWord:for
-0.320006 f previousWord:journey
-0.354515 p previousWord:more
-0.375808 f nextWord:such
-0.451217 f previousWord:all

F. 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 as bias. (Suykens and Vandewalle, 1999)

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

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. (Safavian and Landgrebe, 1991)

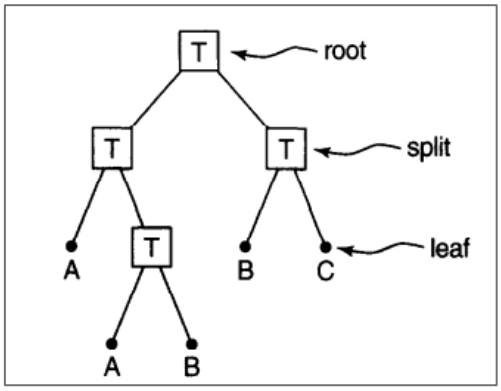


Figure 5. Decision Tree

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.(Safavian and Landgrebe, 1991)

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. (Chen and Guestrin, 2016)

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,

$$y^T{}_i = \varphi(x_i)$$

$$= \sum_{k=1}^{K} f_k(x_i),\ f_k \in F \text{ (Chen and Guestrin, 2016)}$$

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

The boosting algorithm has been optimized using Algorithm 1

Algorithm 1 Boosting Algorithm

Input *Full training set of N examples; maximum ensemble size T; sa size L ¡ N.*

Approach to Boosting

Assign an equal weight of 1/N to all training examples

for i=1 to T **do**

a) based on current weights, randomly sample L examples from trainin without replacements

b)train classifier on this sample

c) identify misclassifed examples

d) increase weights for misclassified examples

Output: ***Final Model based on all classifiers***

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 and proves to output high accuracy, because of it not being dependent on any decision tree, but a bunch, 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. (Cutler et al., 2012)

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. (Saha and Ekbal, 2013)

SMOTE

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 handle this imbalance, Synthetic Minority Over-sampling Technique (SMOTE) approach is adapted for high dimensional binary settings. Generally, used for handling class imbalance. (Chawla et al., 2002)

It is an over sampling technique that synthetically over samples minority classes using novel distance metric approaches.

SMOTE, computes neighbourhood for each minority sample and considers only a subset of the available attributes of the task. (Chawla et al., 2002)

It uses Euclidean, Chebyshev and Manhattan distance metrics and Fisher Score, Mutual Information, Eigenvector and Correlation score as ranking strategies.

For the study, SMOTE over each of our classification algorithms to be able to determine implied aspects of each.

G. Appendix G

It is crucial to understand the fact that the stop words 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 analyse the text semantically. Therefore, in dependency parsing step, the text was used without removing the stop words. Another part of pre-processing 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 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.

H. Appendix H

Type Token Ratio

There are some rules for calculating TTR, which are adapted in this study. These rules include following, (TEMPLIN, 1957)

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.

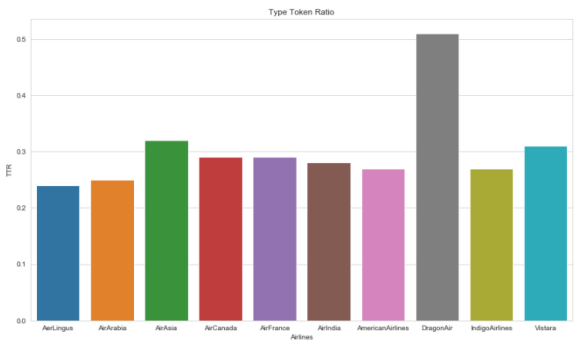


Figure 6. Type-Token Ratio for Airline Ratings

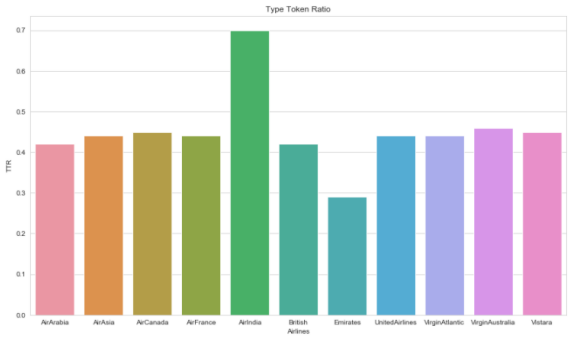


Figure 7 Type-Token Ratio for Trip Advisor

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 law

Zipf's other law states that the number of meanings (m) of a word is the square root of its frequency.(Powers, 1998)

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

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.

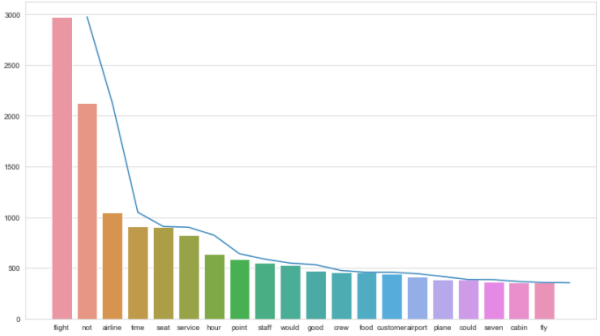


Figure 8. Zipf's distribution

I. 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 accounted, 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. ("The relationship between Precision-Recall and ROC curves | Proceedings of the 23rd international conference on Machine learning," n.d.) 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. ("The relationship between Precision-Recall and ROC curves | Proceedings of the 23rd international conference on Machine learning," n.d.)

Recall: The measure of the correctly identified positive cases from collectively all the actual positive cases. ("The relationship between Precision-Recall and ROC curves | Proceedings of the 23rd international conference on Machine learning," n.d.)

It is significant when the cost of False Negatives is high. Mutually, F1 score is the weighted average of Precision, 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.

In following figures, the scores for best identification model for each algorithm can be found.

1. Conditional Random Field

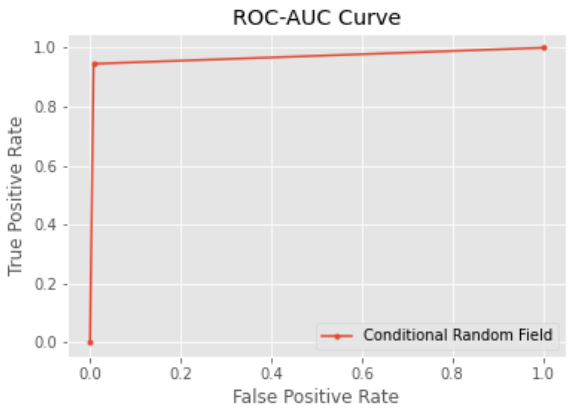


Figure 9. CRF ROC-AUC Curve

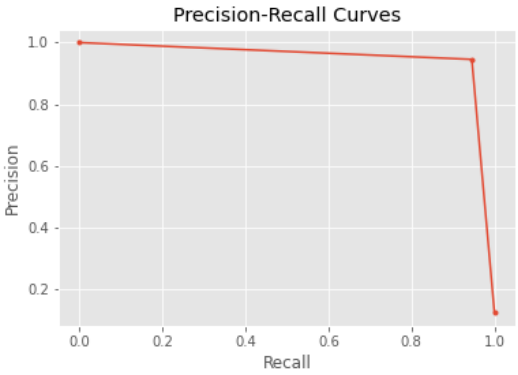


Figure 10. CRF Precision-Recall Curve

```
Metrics for Condtional Random Field Trained on Original Data
 1. ROC AUC Score: 0.9658188763012202
 2. F1 Score: 0.9456094364351245
 3. Average absolute error: 0.01 degrees
 Classification Report is as follows
              precision    recall  f1-score   support

           c       0.95      0.90      0.93      4471
           e       0.92      0.94      0.93      5846
           f       0.97      0.93      0.95     16342
           i       0.88      0.90      0.89      7980
           o       0.96      0.94      0.95     34606
           p       0.97      0.96      0.96      8028
           s       0.90      0.97      0.94     14269
          st       0.94      0.95      0.94     26875

    accuracy                           0.94    118417
   macro avg       0.94      0.94      0.94    118417
weighted avg       0.94      0.94      0.94    118417
```

Figure 11. CRF Classification Report, F1 and Average absolute error scores

2. Support Vector Machines

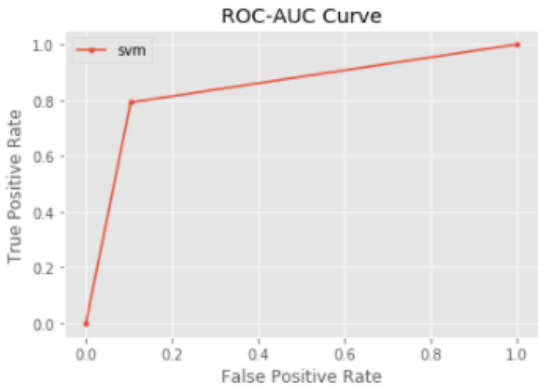


Figure 12. SVM ROC-AUC Curve

```
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


Metrics for svm trained on Original Data
 1. ROC AUC Score: 0.8491974260760425
 2. F1 Score: 0.79204107830552
 3. Average absolute error: 0.14 degrees
```

Figure 13. SVM Classification Report, F1 and Average absolute error scores

3. Decision Tree

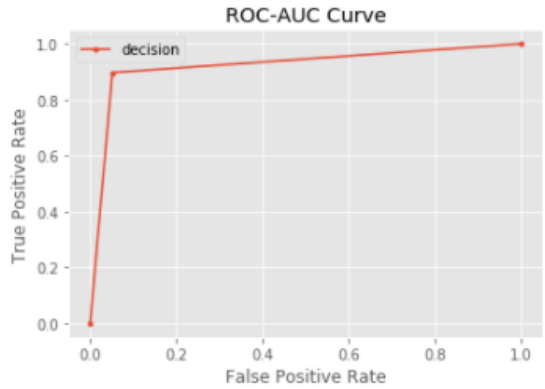


Figure 14. Decision Tree ROC-AUC Curve

```
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


Metrics for decision trained on Original Data
 1. ROC AUC Score: 0.924781802995005
 2. F1 Score: 0.8969616908850726
 3. Average absolute error: 0.07 degrees
```

Figure 15. Decision Tree Classification Report, F1 and Average absolute error scores

4. Random Forest

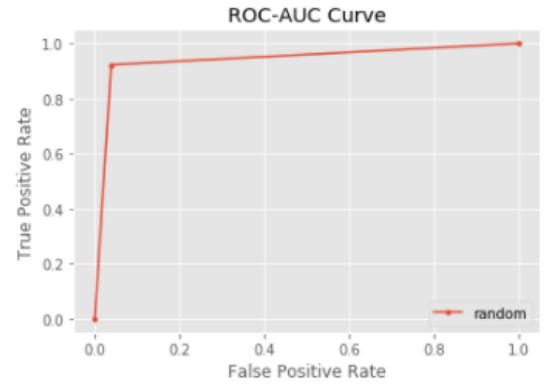


Figure 16. Random Forest ROC-AUC Curve

```
Classification Report is as follows
                  precision    recall  f1-score   support

    food_service       0.91      0.92      0.91       322
      food_taste       0.91      0.92      0.91       288
food_temperature       0.99      0.94      0.96       161

        accuracy                           0.92       771
       macro avg       0.93      0.92      0.93       771
    weighted avg       0.92      0.92      0.92       771


Metrics for random trained on Original Data
 1. ROC AUC Score: 0.9409313163180949
 2. F1 Score: 0.9221789883268483
 3. Average absolute error: 0.05 degrees
```

Figure 17. Random Forest Classification Report, F1 and Average absolute error scores

5. Voting Classifier

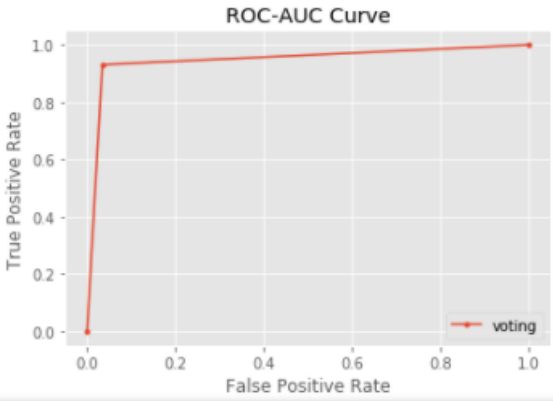


Figure 18. Voting Classifier ROC-AUC Curve

```
Classification Report is as follows
                  precision    recall  f1-score   support

    food_service       0.91      0.91      0.91       310
      food_taste       0.91      0.93      0.92       312
food_temperature       0.99      0.97      0.98       202

        accuracy                           0.93       824
       macro avg       0.94      0.93      0.94       824
    weighted avg       0.93      0.93      0.93       824


Metrics for voting trained on Original Data
 1. ROC AUC Score: 0.948958134665108
 2. F1 Score: 0.9308252427184466
 3. Average absolute error: 0.05 degrees
```

Figure 19. Voting Classifier Classification Report, F1 and Average absolute error scores

6. XG BOOST

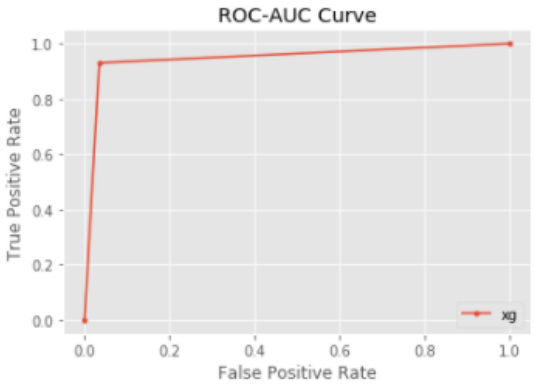
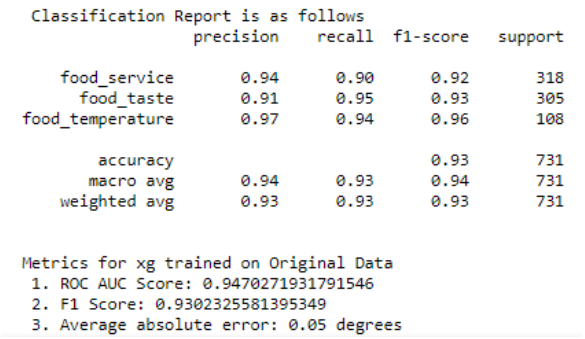


Figure 20. XG BOOST ROC-AUC Curve



```
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


Metrics for xg trained on Original Data
 1. ROC AUC Score: 0.9470271931791546
 2. F1 Score: 0.9302325581395349
 3. Average absolute error: 0.05 degrees
```

Figure 21. XG-BOOST Classification Report, F1 and Average absolute error scores

BIBLIOGRPAHY

Barazza, L., 2017. How does Word2Vec's Skip-Gram work? [WWW Document]. Medium. URL https://becominghuman.ai/how-does-word2vecs-skip-gram-work-f92e0525def4 (accessed 4.23.20).

Bengio, Y., Ducharme, R., Vincent, P., 2001. A Neural Probabilistic Language Model, in: Leen, T.K., Dietterich, T.G., Tresp, V. (Eds.), Advances in Neural Information Processing Systems 13. MIT Press, pp. 932–938.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: Synthetic Minority Over-sampling Technique. jair 16, 321–357. https://doi.org/10.1613/jair.953

Chen, T., Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Presented at the KDD '16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, San Francisco California USA, pp. 785–794. https://doi.org/10.1145/2939672.2939785

Cutler, A., Cutler, D.R., Stevens, J.R., 2012. Random Forests, in: Zhang, C., Ma, Y. (Eds.), Ensemble Machine Learning: Methods and Applications. Springer US, Boston, MA, pp. 157–175. https://doi.org/10.1007/978-1-4419-9326-7_5

Document (Stanford CoreNLP API) [WWW Document], n.d. URL https://nlp.stanford.edu/nlp/javadoc/javanlp-3.5.0/edu/stanford/nlp/dcoref/Document.html (accessed 8.17.20).

Goldberg, Y., Levy, O., 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. arXiv:1402.3722 [cs, stat].

Harris, Z.S., 1954. Distributional Structure. WORD 10, 146–162. https://doi.org/10.1080/00437956.1954.11659520

Lafferty, J., McCallum, A., Pereira, F.C.N., n.d. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data 10.

Majumder, P., Mitra, M., Chaudhuri, B.B., n.d. N-gram: a language independent approach to IR and NLP 7.

McCallum, A., Freitag, D., Pereira, F., n.d. Maximum Entropy Markov Models for Information Extraction and Segmentation 26.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J., 2013. Distributed Representations of Words and Phrases and their Compositionality, in: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 26. Curran Associates, Inc., pp. 3111–3119.

Paltoglou, G., Thelwall, M., 2013. More than Bag-of-Words: Sentence-based Document Representation for Sentiment Analysis, in: Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013. Presented at the RANLP 2013, INCOMA Ltd. Shoumen, BULGARIA, Hissar, Bulgaria, pp. 546–552.

Penn Treebank P.O.S. Tags [WWW Document], n.d. URL https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html (accessed 8.24.20).

Pennington, J., Socher, R., Manning, C., 2014. GloVe: Global Vectors for Word Representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Presented at the EMNLP 2014, Association for Computational Linguistics, Doha, Qatar, pp. 1532–1543. https://doi.org/10.3115/v1/D14-1162

Powers, D.M.W., 1998. Applications and Explanations of Zipf's Law, in: New Methods in Language Processing and Computational Natural Language Learning.

Ratnaparkhi, A., 1996. A Maximum Entropy Model for Part-Of-Speech Tagging, in: Conference on Empirical Methods in Natural Language Processing.

Rosenberg, A., Binkowski, E., 2004. Augmenting the kappa statistic to determine interannotator reliability for multiply labeled data points, in: Proceedings of HLT-NAACL 2004: Short Papers. Presented at the HLT-NAACL 2004, Association for Computational Linguistics, Boston, Massachusetts, USA, pp. 77–80.

Safavian, S.R., Landgrebe, D., 1991. A survey of decision tree classifier methodology. IEEE

Transactions on Systems, Man, and Cybernetics 21, 660–674. https://doi.org/10.1109/21.97458

Saha, S., Ekbal, A., 2013. Combining multiple classifiers using vote based classifier ensemble technique for named entity recognition. Data & Knowledge Engineering, Natural Language for Information Systems: Communicating with Anything, Anywhere in Natural Language 85, 15–39. https://doi.org/10.1016/j.datak.2012.06.003

Smetanin, S., 2018. Google News and Leo Tolstoy: Visualizing Word2Vec Word Embeddings with t-SNE [WWW Document]. Medium. URL https://towardsdatascience.com/google-news-and-leo-tolstoy-visualizing-word2vec-word-embeddings-with-t-sne-11558d8bd4d (accessed 4.23.20).

Suykens, J.A.K., Vandewalle, J., 1999. Least Squares Support Vector Machine Classifiers. Neural Processing Letters 9, 293–300. https://doi.org/10.1023/A:1018628609742

TEMPLIN, M.C., 1957. Certain Language Skills in Children: Their Development and Interrelationships, NED-New edition. ed. University of Minnesota Press. https://doi.org/10.5749/j.ctttv2st

The relationship between Precision-Recall and ROC curves | Proceedings of the 23rd international conference on Machine learning [WWW Document], n.d. URL https://dl.acm.org/doi/abs/10.1145/1143844.1143874?casa_token=5k6e4hFjZhgAAAAA:JrMAFwEeEMTjdPbM0txoFOee59B5RK9Mj2sORe3I04GGYu6g0G2y0pf5LwbdD6hyT7YXtUIQ5x8 (accessed 8.24.20).