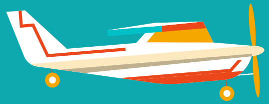
Bon Voyage!
The Stress-Free, Automated Message Explainer (SAME)

Let the robots do the *same* ol' work
so you don't have to!

Contents



01
Introduction

What's the SAME &
what's the difference?

02
Data Exploration

What data was SAME
trained on?

03
Modeling

What's the brain
behind SAME?

04
Conclusion

How can we use it?

There are so many messages in my inbox!

The SIA[1] Customer Service Centre receives thousands of feedback, compliments and complaints every week

1: Singapore Airlines

Help me sort out the messages!

KrisFlyer & PPS Club (KF) | Lounge, Catering, Amenity (LCA) kits

Top 2 most-asked topics

Others

Objectives: SIA management hired us to

(1) Develop a predictive model to **automatically sort messages into the 3 topics**, and

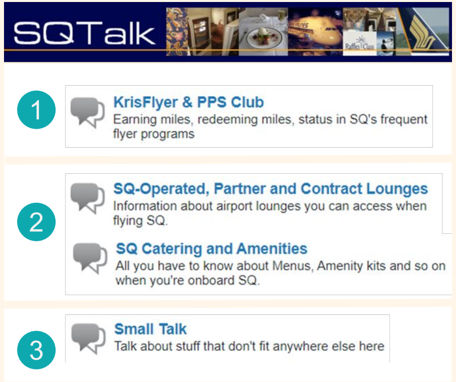
(2) Highlight the **frequently mentioned words and their sentiments** in KF and LCA

SAME:
Always on the move

SAME is a natural language processing model: it can do the same enquiry sorting as we can, but around the clock!

The Approach

Data extraction



16k (40%), 10k (24%), 14k (36%) data rows respectively

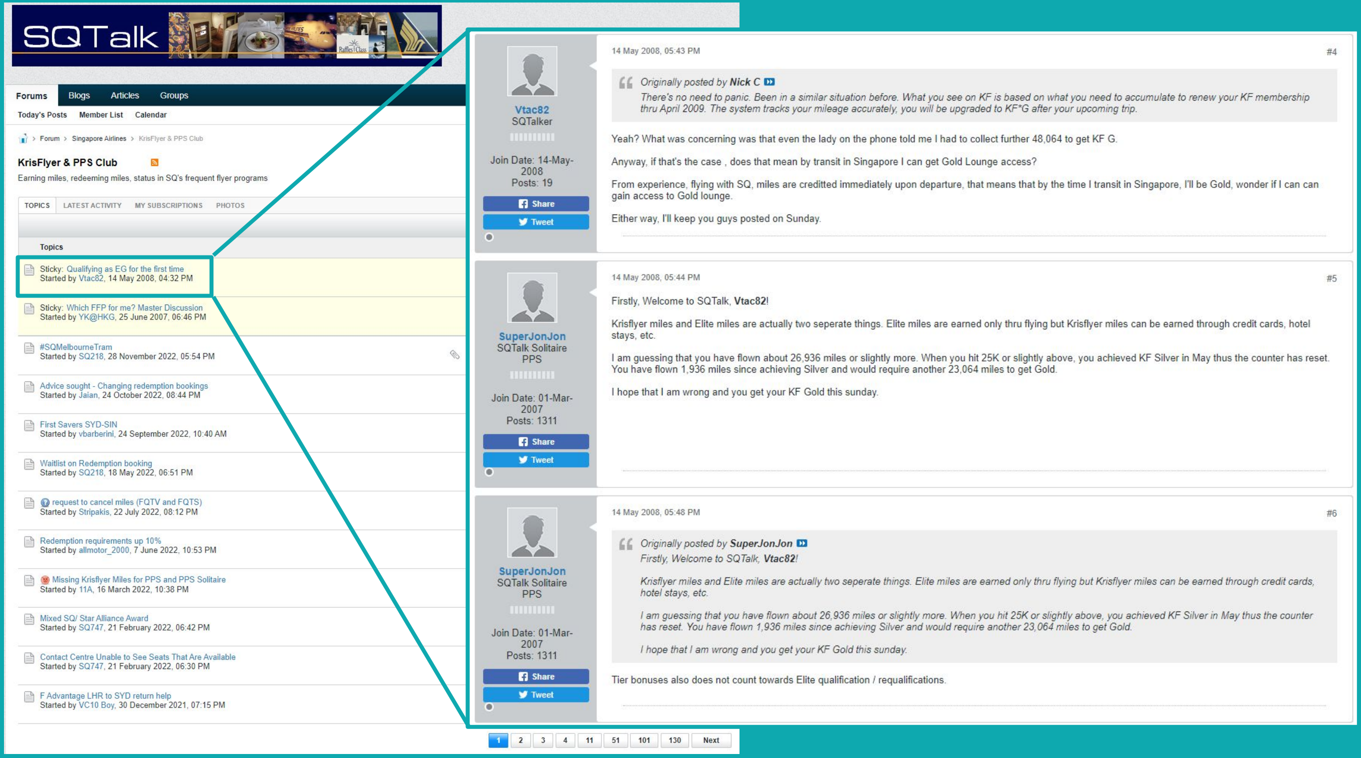
Modelling

Try different **pre-processing, vectoriser, models** & pick the best combination

Sentiment Analysis

Find the **top comments of KF and LCA**, and **analyse the sentiments** of these comments to derive insights

Strap in! Diving into the data



1. **Properly spelled words**
2. **Absence of Singlish**
3. Replies to comments will **duplicate words**
4. **Thread titles are important** as they contain many key words

What's the Model Answer?

1 Lemmatisation vs Stemming vs None:

2 Pre-processing & vectorisation trials:

Bolded – different from baseline

Steps	Baseline	Alternate 1	Alternate 2	Alternate 3
Pre-processing	• Basic data cleaning	• Basic data Cleaning	• Basic data Cleaning • **Remove duplicated sentences**	• Basic data Cleaning • **Remove duplicated sentences**
Vectorisation	CountVectoriser	**TFIDF**	CountVectoriser	**TFIDF**
Model	Multi-Naïve Bayes	Multi-Naïve Bayes	Multi-Naïve Bayes	Multi-Naïve Bayes

$\underbrace{\text{Best performing pre-processing \& vectorisation combination}}$

3 Modelling trials:

Model	Multi-Naïve Bayes	**Random Forest**	**XGBoost**	**SVM**

* The alternate 1, 2, 3 also had 4 additional stopwords (iirc, imo, imho, btw) extracted from a "SQTalk Abbreviations, Slangs, Definitions, Phrases" thread. This had little effect on the model performance.

1 Lemm vs Stem vs None

	Baseline 1	Baseline 2	Baseline 3
Pre-processing	Basic cleaning	- basic cleaning - lemm	- basic cleaning - stem
Vectoriser	CountVectoriser	CountVectoriser	CountVectoriser
Model	Multinomial Naive Bayes	Multinomial Naive Bayes	Multinomial Naive Bayes
macro-average ROC AUC	0.878	0.885	0.887
macro-average f1-score	0.737	0.747	0.752

Remarks:

- **Baseline 3 (remove NA, with stem) performed the best**
- **Macro-average ROC AUC (One vs the Rest)** and **f1-score (labels = KF, LCA)** used as key metrics
  - To compare the model's confidence to distinguish classifications, and focus on minimising false-positives and -negatives for KF and LCA

2 Preprocess, Vectoriser

	Baseline Model (from notebook 2)	Alternate 1 *Best performance*	Alternate 2	Alternate 3
Pre-processing	- Basic cleaning - Stem	- Basic cleaning - Stem	- Basic cleaning - Stem - Remove duplicated sentences	- Basic cleaning - Stem - Remove duplicated sentences
Vectoriser	CountVectoriser	TFIDF	CountVectoriser	TFIDF
Model	Multinomial Naive Bayes	Multinomial Naive Bayes	Multinomial Naive Bayes	Multinomial Naive Bayes
Macro-average ROC AUC	0.887	0.908	0.807	0.819
Macro-average f1-score (kf, lca)	0.752	0.759	0.625	0.630

Best performance: Basic clean, stem, TFIDF

Remarks:

- TFIDF had slightly better performance than CountVectoriser
- **Removing duplicated comments seemed to have an adverse effect** on model performance; this suggests that the comments that people reply to usually have crucial key words in them

Model Comparison

	Model 1 (from notebook 3)	Model 2	Model 3	Model 4 *Best performance*
Vectoriser	TFIDF	TFIDF	TFIDF	TFIDF
Model	Multinomial Naive Bayes	Random Forest	XGBoost	SVM
Macro-average ROC AUC	0.908	0.883	0.925	0.925
Macro-average f1-score (kf, lca)	0.759	0.719	0.807	0.815

Best performance: SVM

Remarks:

- **Model 3 performed the best** with the following params:
  - TfidfVectorizer(max_features=500, stop_words=stem_stopwords)
  - SVC(C=1, gamma=1, probability=True, random_state=42)

- SVM edged out due to its **slightly better f1-score**

Sentiment Analysis: Top Words

KrisFlyer & PPS Club

	count
miles	8899
SQ	8627
posted	7734
Originally	7367
PPS	5771
SIN	4730
flight	3692
KF	3616
just	3547
year	2822

Lounges, Catering, Amenity Kits (LCA)

	count
SIN	7319
lounge	4650
SQ	4337
posted	3819
Originally	3637
The	2213
served	2031
chicken	1928
flight	1872
vegetables	1805

1. Find top words
2. Find sentences that contain these words
3. Sort the sentiment score* by:

Sentiment	Score	Assigned value
Very Negative	Under -0.5	1
Negative	Between -0.5 and -0.1	2
Neutral	Between -0.1 and 0.1	3
Positive	Between 0.1 and 0.5	4
Very Positive	Over 0.5	5

*Using spacytextblob; it is a pipeline component that enables sentiment analysis using the TextBlob library (link)

Sentiment Analysis: Results

Sentiment Score	KrisFlyer 'miles'	KrisFlyer 'PPS'	KrisFlyer 'KF'	LCA 'lounge'	LCA 'SQ'	LCA 'served'
Very negative (1)	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1
Negative (2)	<0.1	<0.1	<0.1	<0.1	0.1	0.1
Neutral (3)	0.3	0.4	0.4	0.4	0.4	0.6
Positive (4)	0.6	0.6	0.6	0.5	0.4	0.3
Very positive (5)	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1
Overall:	Positive	Positive	Positive	Positive	Positive	Positive

* Due to rounding, total may not add up to 1

Remarks:

- Sentiments for comments containing top words are **positive** → **these are areas of strength for SIA**
- Separate deep-dive into these comments can be conducted to find out the reasons why (e.g. good service, good exclusive deals for members, good food served etc.)

Conclusion

Model	**Successfully developed SAME**: a prediction model with macro-average ROC AUC (0.93) and f1-scores (0.82)
Sentiment Analysis	Comments with ‘miles’, ‘PPS’, ‘KF’, ‘lounge’, ‘SQ’, ‘served’ have **largely positive sentiments → areas of strength for SIA**
Next Steps (areas for future improvement)	- **Sort by topic (e.g. KF, LCA, others):** SAME can be used as a backend engine for a chatbot to sort incoming messages - **Sort by type (feedback, complaints, compliments)**: SAME + Sentiment Analysis - **Develop multi-label model** to detect comments with >1 topics (e.g. both KF and LCA) - Find **top words with negative sentiments** to identify and improve on areas of weaknesses

Thanks!

Your work will never be the *same* again

Annex A - Macro Average f1-Score

Label	Per-Class F1 Score	Macro-Averaged F1 Score
Airplane	0.67	$\frac{0.67 + 0.40 + 0.67}{3}$ = **0.58**
Boat	0.40	
Car	0.67	

The macro-averaged F1 score (or macro F1 score) is computed using the arithmetic mean (aka unweighted mean) of all the per-class F1 scores.

In general, if you are working with an imbalanced dataset where all classes are equally important, using the macro average would be a good choice as it treats all classes equally. (source)