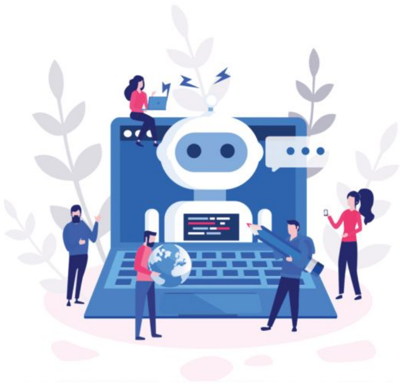
Sentiment Analysis Bot via Emotion AI

An AI-powered CRM Chatbot



Phase 1–2

Lu Liu & Daniel Amin
CSC 74011 - Artificial Intelligence
Fall 2019

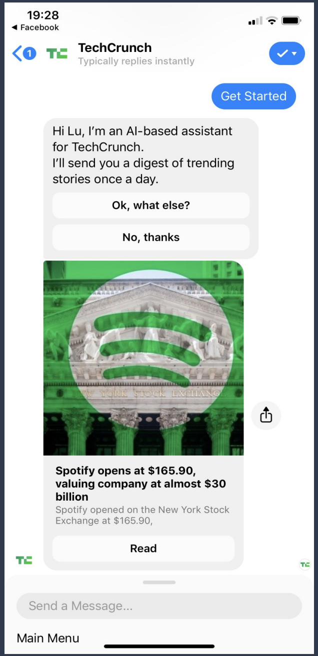
Table of Content
Phase 1

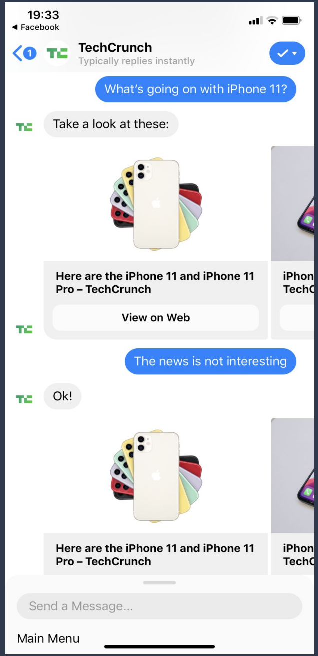
- ❑ Motivation
- ❑ Problem Statement
- ❑ Goal
- ❑ Terminology
- ❑ Related Work
- ❑ Dataset
- ❑ Agent
- ❑ Environment
- ❑ Problem Formulation
- ❑ Search Algorithm
- ❑ Implementation
- ❑ Future Work
- ❑ References

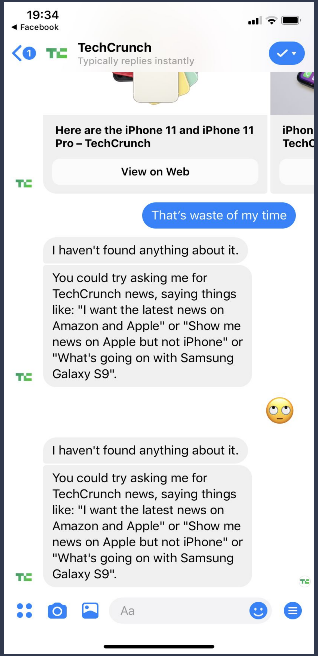
Motivation

Many e-commerce companies utilize online customer service platforms, such as live chat, or messenger bot, to handle customer questions and/or requests in real-time. The concept is ideal and trending.

However, in reality, existing chatbots do not always generate satisfying results. Some of them are lack of intelligence -- the bot respond the customer by predefined actions. While the search of customer need does not match the goal, the bot may repetitively return the same closed result and it neglects customer emotion during the interaction.



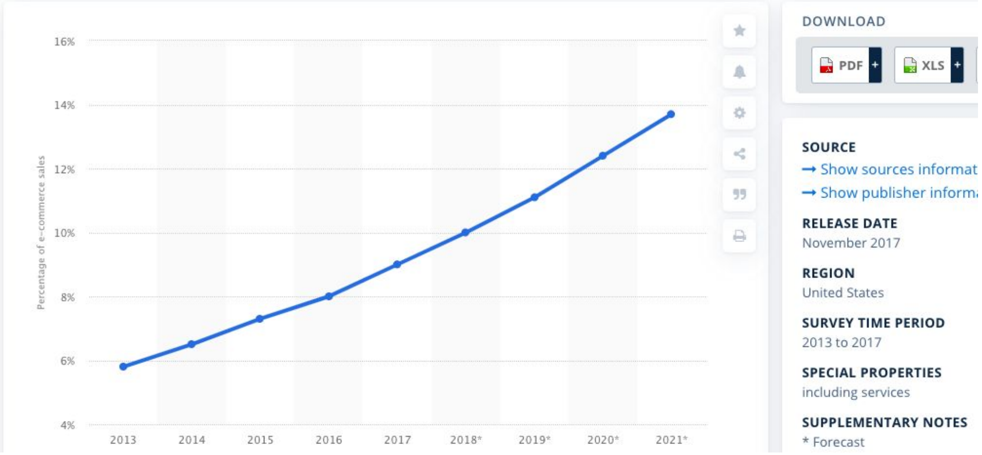


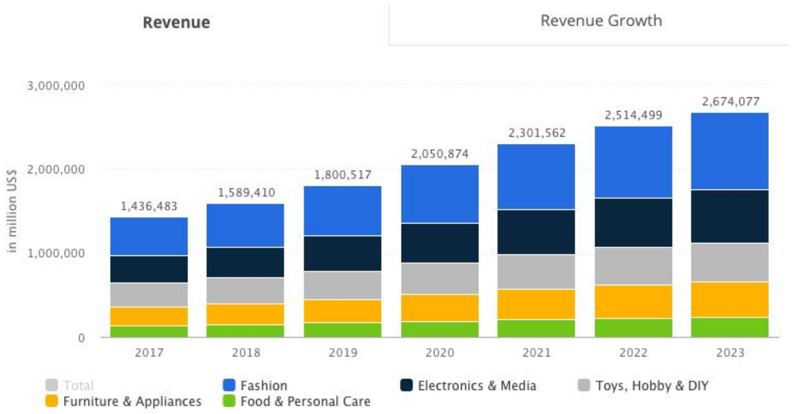


Lu Liu & Daniel Amin. (2019)

★ Revenue in the eCommerce market amounts to US$1,800,517m in 2019.
★ Revenue is expected to show an annual growth rate (CAGR 2019-2023) of 10.4%, resulting in a market volume of US$2,674,077m by 2023.
★ The average revenue per user (ARPU) currently amounts to US$467.33.

E-commerce share of total retail sales in United States from 2013 to 2021





Reference:
[Statista. E-commerce share of total retail sales in United States from 2013 to 2021.]
[Statista. eCommerce - worldwide.]

The World of E-Commerce

Lu Liu & Daniel Amin. (2019)

Terminology

Customer Service/Support: Problem-solving/Solution-focused; Reactive

Customer Success: Opportunity-focused; Proactive

CRM: **Customer Relation Management** is a technology for managing all your company's relationships and interactions with customers and potential customers (Salesforce).

Live Chat/Support: "A Web service that allows businesses to communicate, or chat, in real time with visitors to their Web site" (webopedia.com). The interaction is carried out by a human.

Chatbot: "A computer program designed to simulate conversation with human users, especially over the Internet" (Oxford Dictionaries). The interaction is carried out by a machine.

Terminology (Cont.)

Consumer Packaged Goods (CPG): Items used daily by average consumers that require routine replacement or replenishment, such as food, beverages, clothes, tobacco, makeup, and household products (investopedia.com).

Business-to-Business (B2B): Business that is conducted between companies, rather than between a company and individual consumers (investopedia.com). E.g., Oracle, IBM (enterprise solutions)

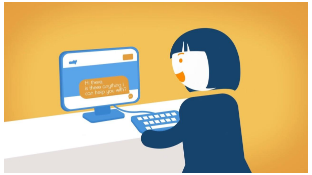
Business-to-Consumer (B2C): The process of selling products and services directly between consumers who are the end-users of its products or services (investopedia.com) E.g., Amazon (except it's own branded products)

Direct-to-Customer (D2C): A low barrier-to-entry eCommerce strategy that allows manufacturers and CPG brands to sell directly to the consumer (coredna.com). E.g., Warby Parker

- Skip retailers or resellers
- Brands sell directly through an online medium
- One type of the B2C business

Small D2C E-commerce Company - Live Chat

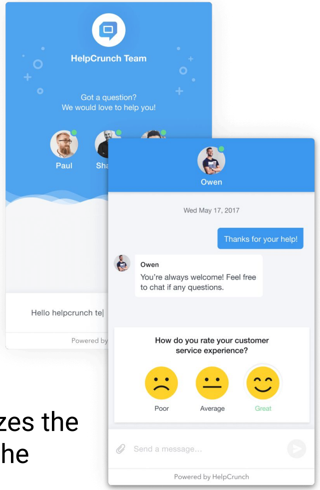
Develop digital text documents for FAQ (Frequently Asked Questions)



Customer representative log on the live chat platform and interact with customers

The rep searches the keywords of the customer message in the saved FAQ document manually to find the closed answers

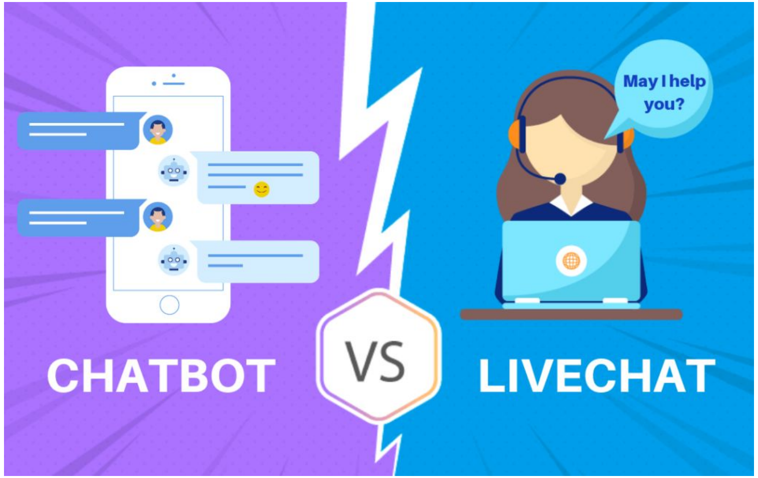
The rep analyzes the sentiment of the message and responds to it accordingly with the most closed answer found



Lu Liu & Daniel Amin. (2019)

PROS:
Efficiency
24/7
Pre-designed system
Multitasking process
Cost-saving

CONS:
Ineffectiveness
Annoying
Cold robot~



PROS:
Better understanding
Engagement
Ability to adapt
Handle exceptions
It's human~

CONS:
Time-consuming
Inefficiency
Inability for complexity
Staffing cost

Why can't we have both?

Problem Statement

Side Effects of Poor Online Customer Service:

- Loss of existing/prospective customers
- Bad brand reputation
- Waste of marketing expense & increased service costs
- Loss of profits

“Emotionless chatbots are taking over customer service – and it’s bad news for consumers” (Polani, The Conversation).

According to CGS study’s annual Global Consumer Customer Service Report, “nearly 50 percent of U.K. respondents and around 40 percent of U.S. respondents said they'd prefer a person” to chatbots (Christopher, Forbes).

Goal

Build a **Sentiment Analysis Bot** that can help the business to analyze customer questions and comments and recognize their emotions so that the bot can:

- respond to customers with the appropriate categories of answers for questions in real-time
- prioritize the events based on the analysis result of customers emotions and make smart decisions in particular scenarios, e.g., send discount code to the disappointed customer, recommend intro video to the new/excited customer, schedule 1-1 in-person call with the mad customer, identify fake complaints and fraud refund/return requests

→ *Emotion AI*

Related Work

Huang, L. & Zhao, K. & Ma, M. **When to Finish? Optimal Beam Search for Neural Text Generation (modulo beam size)**. *Association for Computational Linguistics*.

- Huang et al. present an optimal beam search algorithm for neural text generation, which "will always return the optimal-score complete hypothesis (modulo beam size), and finish as soon as the optimality is established (finishing no later than the baseline) (Association for Computational Linguistics).
- Their "bounded length reward mechanism allows a modified version of the beam search algorithm to remain optimal" (Huang et al.).

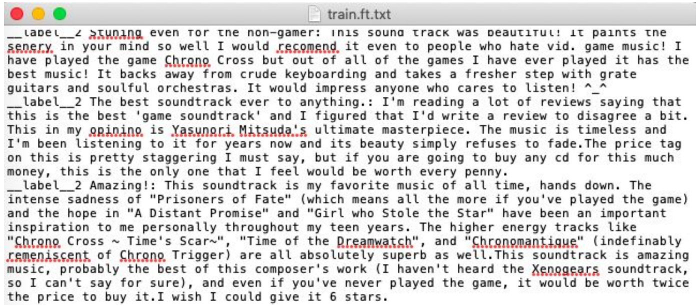
Al-Amir, A. **A Nifty Large-Scale Text Search Algorithm Tutorial**. *Toptal.com*.

- Al-Amir introduces a text search approach via the trie data structure (Toptal.com). He compares the direct approach of looping the search of phrases one by one with a reverse search, which indexes the search terms first and then searches the text body through the index tree.
- His trie approach is more effective than the basis directly approach and it's scalable.
- Our search algorithm in Phase 1 adopts Al-Amir's trie approach.

Datasets (Main & Alternatives)

Text Data (Main):

- Amazon Reviews for Sentiment Analysis - Kaggle (*Source:* *https://www.kaggle.com/bittlingmayer/amazonreviews*)



train.ft.txt

```
__label__2 Stuning even for the non-gamer: This sound track was beautiful! It paints the
senery in your mind so well I would recomend it even to people who hate vid. game music! I
have played the game Chrono Cross but out of all of the games I have ever played it has the
best music! It backs away from crude keyboarding and takes a fresher step with grate
guitars and soulful orchestras. It would impress anyone who cares to listen! ^_^
__label__2 The best soundtrack ever to anything.: I'm reading a lot of reviews saying that
this is the best 'game soundtrack' and I figured that I'd write a review to disagree a bit.
This in my opinino is Yasunori Mitsuda's ultimate masterpiece. The music is timeless and
I'm been listening to it for years now and its beauty simply refuses to fade.The price tag
on this is pretty staggering I must say, but if you are going to buy any cd for this much
money, this is the only one that I feel would be worth every penny.
__label__2 Amazing!: This soundtrack is my favorite music of all time, hands down. The
intense sadness of "Prisoners of Fate" (which means all the more if you've played the game)
and the hope in "A Distant Promise" and "Girl who Stole the Star" have been an important
inspiration to me personally throughout my teen years. The higher energy tracks like
"Chrono Cross ~ Time's Scar~", "Time of the Dreamwatch", and "Chronomantique" (indefinably
remeniscent of Chrono Trigger) are all absolutely superb as well.This soundtrack is amazing
music, probably the best of this composer's work (I haven't heard the Xenogears soundtrack,
so I can't say for sure), and even if you've never played the game, it would be worth twice
the price to buy it.I wish I could give it 6 stars.
```

Alternative Datasets:

- Sentiment Analysis: Emotion in Text - Kaggle *(Source: https://www.kaggle.com/c/sa-emotions/data)*
- Sentiment Analysis in Text - data.world *(Source: https://data.world/crowdflower/sentiment-analysis-in-text)*
- Sarcasm on Reddit - Kaggle *(Source: https://www.kaggle.com/danofer/sarcasm)*

Datasets (Cont.)

Tableau Emotion Data:

- Emotions Sensor Data Set - Words Classified Statistically Into 7 Basic Emotions (*Source: https://www.kaggle.com/iwilldoit/emotions-sensor-data-set*)

Andbrain_DataSet.csv (113.37 KB)   8 of 8 columns   Views

	word	disgust	surprise	neutral	anger	sad
	1104 unique values	0 – 0.14	0 – 0.14	0 – 0.04	0 – 0.14	0
1	ability	0.004464286	0.047831636	6.3775515E-4	0.023596939	
2	able	1.7322194E-5	1.823925E-4	4.0859997E-4	1.762788E-4	
3	abuse	5.317264E-4	1.7724212E-4	1.7724212E-4	0.13736264	
4	academy	0.0071428576	0.021428574	0.0071428576	0.0071428576	
5	accept	0.008270677	0.0067669176	7.5187976E-4	0.048872184	
6	acceptance	0.002747253	0.008241759	0.002747253	0.013736265	
7	accounting	0.017857144	0.017857144	0.017857144	0.017857144	
8	accuracy	0.035714287	0.107142866	0.035714287	0.035714287	
9	achieve	0.0013824885	0.004147466	4.608295E-4	0.0050691245	
10	acid	0.017857144	0.017857144	0.017857144	0.017857144	

Lu Liu & Daniel Amin. (2019)

Agent

Application Type: CRM Chatbot

Agent Type: Learning Agent

Percepts: Input Words

Actions: Analyze customer sentiment; react to customer questions and comments; assign tasks to specialists; resolve customer problem

Goals: Convert negative sentiment to positive (i.e., transform anger to happy) and maximize customer satisfaction

Environment: Online Customers

The Learning Model:

- **Learning Problem**: Improve over task T with respect to performance measure P, based on experience E.
- **Task (T)**: Analyze customer sentiment for customer success
- **Performance Measure (P)**: % of customer comments correctly classified and responded; maximized customer satisfaction & positive sentiment
- **Experience (E)**: Pre-analyzed customer comments

Lu Liu & Daniel Amin. (2019)

Environment

Environment Properties:

- Discrete; Partially Observable; Dynamic; Single Agent; Inaccessible; Non-deterministic; Non-episodic

Task Environment (PEAS) -- The first step of intelligent agent design

- **Performance Measure**: % of customer comments correctly classified and responded; maximized customer satisfaction & positive sentiment
- **Environment**: Online Customers
- **Actuators**: A display to interact with customers
- **Sensors**: Keyboard

Problem Formulation

Initial State: Customer's first question/comment and the corresponding emotion (greetings are not counted here)

Goal State: Customer's needs are fulfilled and the final emotion is happy

States: Various topics/categories and emotions

Actions: All possible actions that the chatbot can perform to provide relevant info to the customer that leads to the goal

- E.g., Analyze customer sentiment; react to customer questions and comments; assign tasks to specialists; resolve customer problem

Goal Test: Does the customer get what he/she wants? Is the customer happy?

Problem Formulation

Solution: Problem Solving by Search and Sentiment Analysis in This Application

- Retrieval-based Model: Pick up the most appropriate response(s) from a set of predefined answers/actions and a ranking model of sentiments
- If none of the possible predefined answers/actions can achieve the goal directly, then return a sequence of action that leads to the final goal.

Phase 1

Using Text (Phrase) Search to associate keywords from customer input with specific topics/categories, e.g. order, shipping, QA, etc.

Search Algorithm

Steps for **Text Search (Phase Search)** through **Trie**:

1. Create a list of phrases representing specific actions and rules that associate with the corresponding topics/categories. E.g.,
    a. **Search Terms**: order status, cancel an order, cancel my order, wrong order → **Topic**: Order
    b. **Search Terms**: can i exchange, return my order, return the product, waste of money → **Topic**: Returns and Refunds
    c. **Search Terms**: talk to someone, speak with someone, customer service number, customer representative, this is ridiculous → **Topic**: 1-to-1 Support
2. Index the list of phrases into a trie
3. Search the text body through the trie
    a. Trie Pointer -- the Start Node/Root
    b. Word Pointer -- the Word Node after the Start Node
4. Incorporate with Search Algorithm(s)

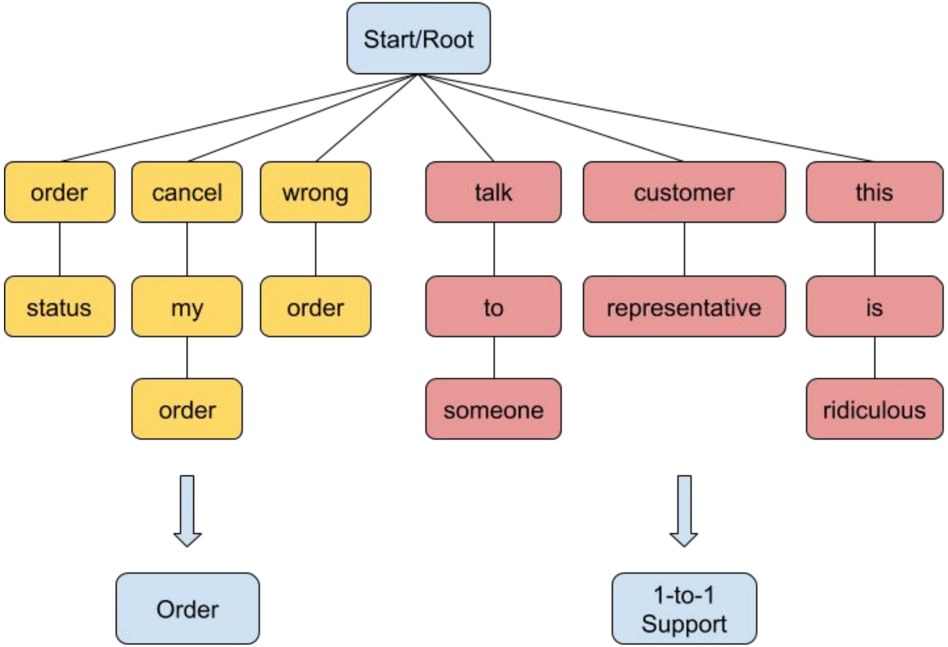
Lu Liu & Daniel Amin. (2019)

Search Algorithm

Search Algorithms in AI Chatbot:

Uninformed Search

- **Depth-First Search** for phrase search (Phase 1)



Index Trie of Search Terms (Phrases)

- Extend path by word:

Lu Liu & Daniel Amin. (2019)

Implementation – Trie

```
[ ]  class Trie:
         head = {}

         def add(self, phrases, label):
             cur = self.head
             for word in phrases.split():
                 if word not in cur:
                     cur[word] = {}
                 cur = cur[word]
             cur['*'] = 'Label' + label

         def search(self, phrases, label):
             cur = self.head
             for word in phrases.split():
                 if word not in cur:
                     return False
                 cur = cur[word]

             if 'Label' + label in cur['*']:
                 return True
             else:
                 return False
```

```
[ ]  dictionary = Trie()

     dictionary.add("cant login", "Account")
     dictionary.add("cant log in", "Account")
     dictionary.add("cannot log in", "Account")
     dictionary.add("cannot login", "Account")
     dictionary.add("reset my password", "Account")
     dictionary.add("reset password", "Account")
     dictionary.add("account is locked", "Account")
     dictionary.add("delete my account", "Account")

     dictionary.add("order", "Order")
     dictionary.add("order status", "Order")
     dictionary.add("cancel an order", "Order")
     dictionary.add("cancel my order", "Order")
     dictionary.add("wrong order", "Order")
     dictionary.add("hold an order", "Order")
     dictionary.add("still processing", "Order")
     dictionary.add("change shipping address", "Order")
     dictionary.add("wrong shipping address", "Order")
```


Lu Liu & Daniel Amin. (2019)

Implementation - Sample Text

```
Great CD: My lovely Pat has one of the GREAT voices of her generation. I have listened to this CD for YEARS and I still LOVE IT. When I'm in a good mood it makes me feel bette
One of the best game music soundtracks - for a game I didn't really play: Despite the fact that I have only played a small portion of the game, the music I heard (plus the con
Batteries died within a year ...: I bought this charger in Jul 2003 and it worked OK for a while. The design is nice and convenient. However, after about a year, the batteries
works fine, but Maha Energy is better: Check out Maha Energy's website. Their Powerex MH-C204F charger works in 100 minutes for rapid charge, with option for slower charge (be
Great for the non-audiophile: Reviewed quite a bit of the combo players and was hesitant due to unfavorable reviews and size of machines. I am weaning off my VHS collection, b
DVD Player crapped out after one year: I also began having the incorrect disc problems that I've read about on here. The VCR still works, but hte DVD side is useless. I unders
Incorrect Disc: I love the style of this, but after a couple years, the DVD is giving me problems. It doesn't even work anymore and I use my broken PS2 Now. I wouldn't recomme
DVD menu select problems: I cannot scroll through a DVD menu that is set up vertically. The triangle keys will only select horizontally. So I cannot select anything on most DV
Unique Weird Orientalia from the 1930's: Exotic tales of the Orient from the 1930's. "Dr Shen Fu", a Weird Tales magazine reprint, is about the elixir of life that grants immo
Not an "ultimate guide": Firstly,I enjoyed the format and tone of the book (how the author addressed the reader). However, I did not feel that she imparted any insider secrets
```

Implementation - Search

```
with open('sample.txt') as f:
    content = f.readlines()
content = [x.strip() for x in content]

import nltk
import string
from nltk.corpus import stopwords

stop = stopwords.words('english')
w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()

def lemmatize_text(text):
    return [lemmatizer.lemmatize(w) for w in w_tokenizer.tokenize(text)]



x = 1

for line in content:
  line = line.lower()
  line = line.replace('-', ' ')
  #line = line.split(' ')
  #line = line.apply(lambda x: [item for item in x if item not in stop])
  #line = line.apply(', '.join)
  #line = line.replace('[{}]'.format(string.punctuation), '')
  #line = line.apply(lemmatize_text)
  #line = line.apply(', '.join)
  line = line.replace('[{}]'.format(string.punctuation), '')

  print('Comment ' + str(x))
  searchClass(line)
  x = x + 1
```

```
Comment 1
Comment 2
Comment 3
Comment 4
Comment 5
Comment 6
Comment 7
Comment 8
Comment 9
Comment 10
Comment 11
LabelRec
found:  new to
```

Future Work

Phase 2:

Sentiment Analysis of Amazon Reviews via Classification

Phase 3:

Implementation of Classifier with Text Search Chatbot

References

- Al-Amir, A. A Nifty Large-Scale Text Search Algorithm Tutorial. Retrieved from https://www.toptal.com/algorithms/needle-in-a-haystack-a-nifty-large-scale-text-search-algorithm
- Elliott, C. (2018). Chatbots Are Killing Customer Service. Here's Why. Forbes. Retrieved from https://www.forbes.com/sites/christopherelliott/2018/08/27/chatbots-are-killing-customer-service-heres-why/#7872893a13c5
- Polani, D. (2017). Emotionless chatbots are taking over customer service – and it's bad news for consumers. The Conversation. Retrieved from http://theconversation.com/emotionless-chatbots-are-taking-over-customer-service-and-its-bad-news-for-consumers-82962
- Huang, L. & Zhao, K. & Ma, M. When to Finish? Optimal Beam Search for Neural Text Generation (modulo beam size). Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/D17-1227
- IE University. How to use AI machine learning for brand sentiment analysis. Retrieved from https://drivinginnovation.ie.edu/how-to-use-ai-machine-learning-for-brand-sentiment-analysis/
- minsuk-heo. coding_interview. Github. Retrieved from https://github.com/minsuk-heo/coding_interview/blob/master/trie.ipynb
- How to read a file line-by-line into a list? Stack Overflow. Retrieved from https://stackoverflow.com/questions/3277503/how-to-read-a-file-line-by-line-into-a-list

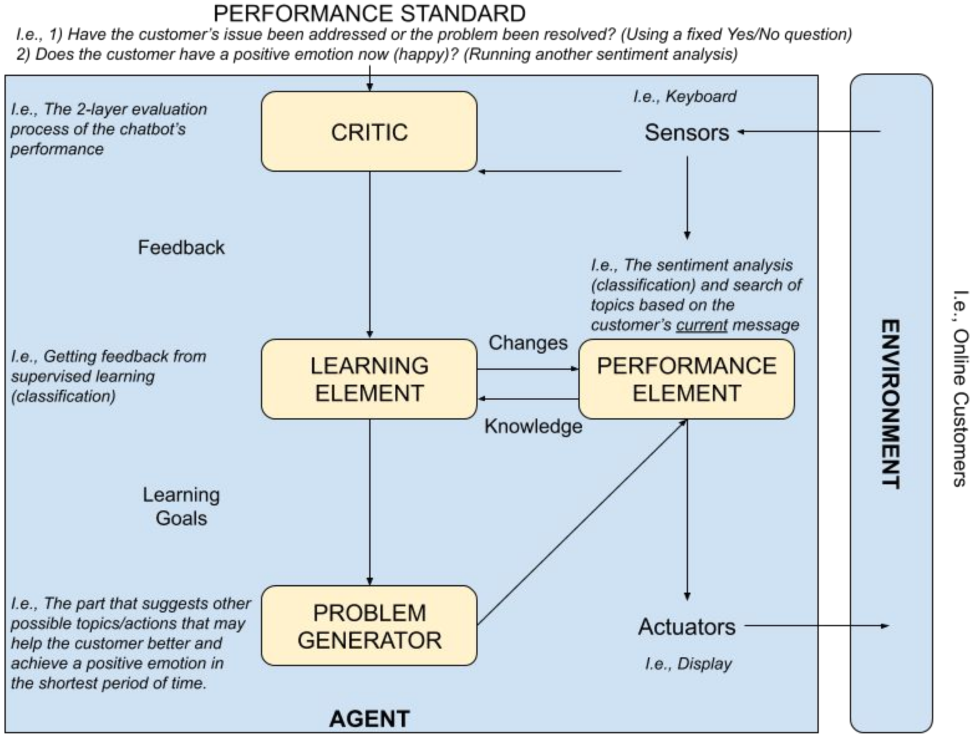
Phase 2

Sentiment Analysis via various algorithms

Table of Content
Phase 2

- ❑ Learning Agent Model
- ❑ Inductive Learning
- ❑ Python Libraries/Modules
- ❑ Algorithms
  - ❑ Bag-of-Word
  - ❑ Naive Bayes Classification
  - ❑ K-means Clustering
  - ❑ Logistic Regression
  - ❑ Support Vector Machine
  - ❑ Random Forest Classification
  - ❑ Word2Vec
  - ❑ Deep Neural Net
  - ❑ CNN
  - ❑ Emotion Lexicon
- ❑ Conclusion
- ❑ References

Learning Agent Model



References: Agents that Learn – UWA
http://teaching.csse.uwa.edu.au/units/CITS4211/Lectures/wk5.pdf

Lu Liu & Daniel Amin. (2019)

Inductive Learning

Inductive Learning: System tries to induce a "general rule" from a set of observed instances.

Supervised Learning: learning algorithm is given the correct value of the function for particular inputs, and changes its representation of the function to try to match the information provided by the feedback.

An example is a pair (x, f(x)), where x is the input and f(x) is the output of the function applied to x.

-- *from Professor Claire Cardie's Lecture Slide CS472 – Machine Learning 4 - Cornell University CS 4740 - Introduction to Natural Language Processing*

Sentiment Analysis:

- The x is the properties of the online customer.
- The f(x) is the sentiment of the customer.

References:
Basic Concepts in Machine Learning https://machinelearningmastery.com/basic-concepts-in-machine-learning/
https://www.cs.cornell.edu/courses/cs4740/2012sp/lectures/ml-basics-ai-lecture-4pp.pdf

Python Libraries/Modules

- Pandas
- Numpy
- Re
- Matplotlib
- Seaborn
- Tensorflow
- Gensim

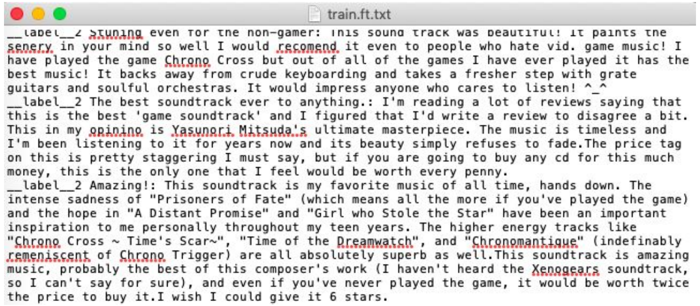
- NLTK
  - punkt
  - stopwords
  - wordnet
    - PorterStemmer
    - WordNetLemmatizer
  - vader_lexicon
    - SentimentIntensityAnalyzer

- Scikit Learn
  - CountVectorizer
  - TfidfVectorizer
  - train_test_split
  - GaussianNB
  - svm
  - KMeans
  - RandomForestClassifier
  - LinearRegression
  - LogisticRegression
  - confusion_matrix
  - accuracy_score

Selected Dataset Recap

Amazon Reviews for Sentiment Analysis - Kaggle (*Source: https://www.kaggle.com/bittlingmayer/amazonreviews*)

- Over millions of documents/reviews
- Text File
- Pre-labeled:
  - "**__label__1**" represents **negative** reviews with 1-2 stars
  - "**__label__2**" indicates **positive** reviews of 4-5 stars
  - **Problem**: 3-star reviews are considered as **neutral** so they are not included in the original dataset.
- Size:
  - Original Training Dataset: 1.6 GB
  - Original Testing Dataset: 177.4 MB


train.ft.txt

__label__2 Stuning even for the non-gamer: This sound track was beautiful! It paints the senery in your mind so well I would recomend it even to people who hate vid. game music! I have played the game Chrono Cross but out of all of the games I have ever played it has the best music! It backs away from crude keyboarding and takes a fresher step with grate guitars and soulful orchestras. It would impress anyone who cares to listen! ^_^
__label__2 The best soundtrack ever to anything.: I'm reading a lot of reviews saying that this is the best 'game soundtrack' and I figured that I'd write a review to disagree a bit. This in my opinino is Yasunori Mitsuda's ultimate masterpiece. The music is timeless and I'm been listening to it for years now and its beauty simply refuses to fade.The price tag on this is pretty staggering I must say, but if you are going to buy any cd for this much money, this is the only one that I feel would be worth every penny.
__label__2 Amazing!: This soundtrack is my favorite music of all time, hands down. The intense sadness of "Prisoners of Fate" (which means all the more if you've played the game) and the hope in "A Distant Promise" and "Girl who Stole the Star" have been an important inspiration to me personally throughout my teen years. The higher energy tracks like "Chrono Cross ~ Time's Scar~", "Time of the Dreamwatch", and "Chronomantique" (indefinably remeniscent of Chrono Trigger) are all absolutely superb as well.This soundtrack is amazing music, probably the best of this composer's work (I haven't heard the Xenogears soundtrack, so I can't say for sure), and even if you've never played the game, it would be worth twice the price to buy it.I wish I could give it 6 stars.

Lu Liu & Daniel Amin. (2019)

Initial Approaches

Training Data: 120K reviews, Testing Data: 40K reviews

- BOW
- Naive Bayes

Training Data: 800 reviews, Testing Data: 200 reviews

- K-means
- Logistic Regression
- SVM
- Random Forest

Labels: “1” for Negative Reviews, “2” for Positive Reviews

Lu Liu & Daniel Amin. (2019)

Bag-of-Words: Implementation

	text	label
844632	Didn't connect: If there were two subjects jus...	1
2053249	A delicious Vegan cookbook. Finally.: Finally ...	2
992772	Absolutely wonderful!: The writing in this boo...	2
950535	contains some basic info: I purchased this boo...	1
3165074	I don't get the hype: After finish reading thi...	1

```
stop = stopwords.words('english')
w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()

def lemmatize_text(text):
    return [lemmatizer.lemmatize(w) for w in w_tokenizer.tokenize(text)]

#lowercase and remove punctuation, remove stopwords
df['text'] = df['text'].str.lower()
df['text'] = df['text'].str.replace('-', ' ')
df['text'] = df['text'].str.split(' ')
df['text'] = df['text'].apply(lambda x: [item for item in x if item not in stop])
df['text'] = df['text'].apply(', '.join)
df['text'] = df['text'].str.replace('[{}]'.format(string.punctuation), '')
df['text'] = df['text'].apply(lemmatize_text)
df['text'] = df['text'].apply(', '.join)
df['text'] = df['text'].str.replace('[{}]'.format(string.punctuation), '')
df['text'] = df['text'].str.replace('\\', ' ')
```

	text	label
844632	connect two subject ripe satirical take down w...	1
2053249	delicious vegan cookbook finally finally vegan...	2
992772	absolutely wonderful writing book descriptive ...	2
950535	contains basic info purchased book hoping woul...	1
3165074	get hype finish reading book hard time underst...	1

Lu Liu & Daniel Amin. (2019)

Bag-of-Words: Result

```
df_1 = df.loc[df['label'] == 1]
df_1_count = df_1.text.str.split(expand=True).stack().value_counts()
df_1_count.head()
```

```
book     35486
one      24148
like     18236
would    17047
it       15431
dtype: int64
```

```
df_2 = df.loc[df['label'] == 2]
df_2_count = df_2.text.str.split(expand=True).stack().value_counts()
```

```
df_2_count.head()
```

```
book     38125
great    27342
one      23244
good     20025
like     16599
dtype: int64
```


Lu Liu & Daniel Amin. (2019)

Naive Bayes Classification: Implementation

```
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(X_train_tf, df.label)

string1 = 'Exciting action gentle romance perfect movie', 'Stolen track deborah cox never got paid'
string1 = model_vect.transform(string1)
print(string1)
print()
string1 = tf_transformer.transform(string1)
print(string1)
predicted = clf.predict(string1)
print()
print(predicted)
```

```
string1 = 'Exciting action gentle romance perfect movie', 'Stolen track deborah cox never got paid'
string1 = model_vect.transform(string1)
print(string1)
print()
string1 = tf_transformer.transform(string1)
print(string1)
predicted = clf.predict(string1)
print()
print(predicted)
```

```
  (0, 7836)	1
  (0, 59664)	1
  (0, 69986)	1
  (0, 108782)	1
  (0, 122372)	1
  (0, 140452)	1
  (1, 41031)	1
  (1, 44720)	1
  (1, 72132)	1
  (1, 112189)	1
  (1, 119979)	1
  (1, 156382)	1
  (1, 168235)	1

  (0, 7836)	0.4082482904638631
  (0, 59664)	0.4082482904638631
  (0, 69986)	0.4082482904638631
  (0, 108782)	0.4082482904638631
  (0, 122372)	0.4082482904638631
  (0, 140452)	0.4082482904638631
  (1, 41031)	0.3779644730092272
  (1, 44720)	0.3779644730092272
  (1, 72132)	0.3779644730092272
  (1, 112189)	0.3779644730092272
  (1, 119979)	0.3779644730092272
  (1, 156382)	0.3779644730092272
  (1, 168235)	0.3779644730092272

[2 1]
```

Naive Bayes Classification: Result

	text	label	prediction
0	great cd lovely pat one great voice generation...	2	2
1	one best game music soundtrack game really pla...	2	2
2	battery died within year bought charger jul 20...	1	1
3	work fine maha energy better check maha energy...	2	1
4	great non audiophile reviewed quite bit combo ...	2	2

```
count_true = 0
false_pos = 0
false_neg = 0

for index, row in df.iterrows():
    if row['label'] == row['prediction']:
        count_true = count_true + 1
    elif row['label'] == 1 and row['prediction'] == 2:
        false_pos = false_pos + 1
    elif row['label'] == 2 and row['prediction'] == 1:
        false_neg = false_neg + 1

print("Accuracy on test set: " + str(count_true/len(df)))
print("False pos: " + str(false_pos/len(df)))
print("False neg: " + str(false_neg/len(df)))
```

```
Accuracy on test set: 0.8507903730240675
False pos: 0.06640733398166504
False neg: 0.08280229299426752
```

Accuracy for Naive Bayes Classifier: 85.1%

→ This result was based on Daniel's 120,000 review from the original training dataset.

Train & Test based on 1,000 Reviews

BOW with Count Vectorizer:

```
[ ] from sklearn.feature_extraction.text import CountVectorizer
```

```
[ ] # Define the count vectorizer:
    countVect = CountVectorizer()  # Build a vocabulary from all words in the review corpus

    reviewArray = countVect.fit_transform(reviewCorpus).toarray()  # Count the number of times a word from vocabulary appears in each sentence of

    labelArray = df1.iloc[: , 0].values
```

```
[ ] # Show the total number of features:
    len(countVect.get_feature_names())

    8208
```

```
[ ] # Print all features:
    print(countVect.get_feature_names())

    ['aa', 'aaarrrggghhh', 'abandon', 'abarca', 'abbreviated', 'abduction', 'ability', 'abit', 'able', 'aboard', 'aboration', 'abound', 'abounds',
```

```
[ ] # Print the number of times each word appears in each document (review):
    print(reviewArray)
```

Because the number of words in the vocabulary is way much larger than the number of features, many word does not existi in most of the review, which is shown as "0" appearence.

Split The Training and Testing Datasets:

```
[ ] # Import train_test_split from Sciket Learn for splitting the dataset:
    from sklearn.model_selection import train_test_split
```

```
[ ] reviewArray_train, reviewArray_test, labelArray_train, labelArray_test = train_test_split(
        tfidf_reviewArray, labelArray, test_size = 0.2, random_state = 234)
```

```
[90] df1['label'].value_counts()

    __label__2    502
    __label__1    498
    Name: label, dtype: int64
```

→ Balanced Sample Data

```
[ ] # Print the first 10 word counts from the vectorization mapping of the first sentence of review:
    list(zip(reviewArray[0], countVect.get_feature_names()))[ : 10]

    [(0, 'aa'),
     (0, 'aaarrrggghhh'),
     (0, 'abandon'),
     (0, 'abarca'),
     (0, 'abbreviated'),
     (0, 'abduction'),
     (0, 'ability'),
     (0, 'abit'),
     (0, 'able'),
     (0, 'aboard')]
```

```
[ ] # if isinstance(reviewArray, np.ndarray):
    #    print("It's a Numpy array.")

    # if isinstance(labels, np.ndarray):
    #    print("It's a Numpy array.")
```

```
[ ] # Replace the original labbels with strings of numbers:
    labelArray = np.where(labelArray == "__label__1", "1", labelArray)  # Use "1" for negative reviews (1-2 Starts)
    labelArray = np.where(labelArray == "__label__2", "2", labelArray)  # Use "2" for postive reviews (4-5 Starts)
```

```
 the Numpy array of strings to integers:
= labelArray.astype(int)

Array.dtype)
```

→ Lu's experiment was based on a subset of the original training dataset, which contains 1,000 amazon review split in 80/20.

Lu Liu & Daniel Amin. (2019)

K-means Clustering: Implementation

Method: **sklearn.feature_extraction.text.TfidfVectorizer**

```
[41] from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.cluster import KMeans
```

```
[42] # Define the Tfidf vectorizer:
     tfidfVect = TfidfVectorizer()  # Build a matrix of TF-IDF features from all words in the review corpus
     tfidf_reviewArray = tfidfVect.fit_transform(reviewCorpus).toarray()

     tfidf_reviewDF = pd.DataFrame(tfidf_reviewArray, columns = tfidfVect.get_feature_names())
     tfidf_reviewDF
```

	aa	aaarrrggghhh	abandon	abarca	abbreviated	abduction	ability	abit	able	aboard	aboration	abound	abounds
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...

```
[43] # Show the total number of features:
     len(tfidfVect.get_feature_names())
```

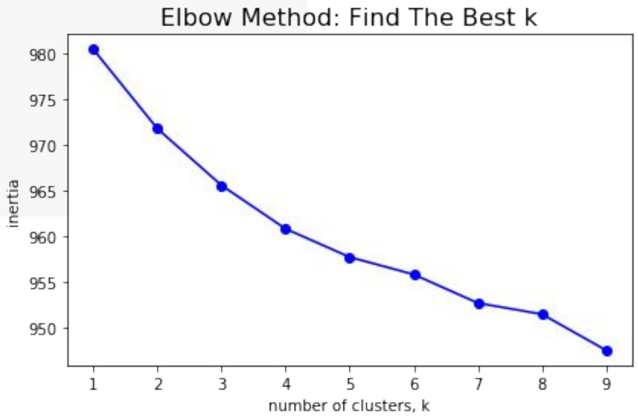
```
8208
```


Lu Liu & Daniel Amin. (2019)

K-means Clustering: Implementation

- Find the best *k*:

```
[44] # Find the best number of clusters:
     ks = range(1, 10)   # Create a sequence of numbers from 1 to 9.
     inertias = []

     for k in ks:
       model = KMeans(n_clusters = k)
       # Select the first 2 PCs by calling .iloc[] on the dataframe:
       # .iloc[] is primarily integer position based (from 0 to length-1 of the axis), or you can use the index with : directly.
       model.fit(tfidf_reviewDF.iloc[:, :])  # Select all columns/features of BOW.
       inertias.append(model.inertia_)

     plt.plot(ks, inertias, '-o', color='blue')
     plt.xlabel('number of clusters, k')
     plt.ylabel('inertia')
     plt.xticks(ks)
     plt.title("Elbow Method: Find The Best k", fontsize = 16)

     plt.tight_layout()
```



Lu Liu & Daniel Amin. (2019)

K-means Clustering: Result

K-means of all features (words):

```
[46] k = 5
     kmModel = KMeans(n_clusters = k, init = 'k-means++', max_iter = 100, n_init = 1, random_state = 2345)
     kmModel.fit(tfidf_reviewArray)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=100,
       n_clusters=5, n_init=1, n_jobs=None, precompute_distances='auto',
       random_state=2345, tol=0.0001, verbose=0)
```

```
[47] print("Top 10 terms per Cluster: ")
     order_centroids = kmModel.cluster_centers_.argsort()[:, ::-1]
     terms = tfidfVect.get_feature_names()

     for i in range(k):
       top_term_words = [terms[ind] for ind in order_centroids[i, :10]]
       print("Cluster {}: {}".format(i, ' '.join(top_term_words)))
```

```
Top 10 terms per Cluster:
Cluster 0: product good work sony charger nice battery price power great
Cluster 1: great one work game would time get well money film
Cluster 2: book read reading story one author like time would great
Cluster 3: cd album music song one band sound like heard track
Cluster 4: movie film time bad great watch good story plot make
```

"great" and "bad" will be the top two features to be used in the following visualation.

```
[51] # Predict a random sentence:
     test = tfidfVect.transform(["love the product but shipping was too slow"])
     kmPred = kmModel.predict(test)
     print("Cluster: ", kmPred)
```

```
Cluster:  [0]
```

```
[50] kmResults = pd.DataFrame()
     kmResults['review'] = reviewCorpus
     kmResults['cluster'] = kmModel.labels_
     kmResults
```

	review	cluster
0	great cd lovely pat one great voice generation...	3
1	one best game music soundtrack game really pla...	3
2	battery died within year bought charger jul wo...	0
3	work fine maha energy better check maha energy...	0
4	great non audiophile reviewed quite bit combo ...	0
...	...	...
995	borinmg dumb waste time glory old time movie t...	4
996	best film year one best film ever made god mon...	1
997	see movie ian mckellen performance god monster...	4
998	best screenplay stability one recent film anti...	1
999	tree arrived bent poorly packed manufacturer p...	1

1000 rows × 2 columns

Lu Liu & Daniel Amin. (2019)

K-means Clustering: Visualization

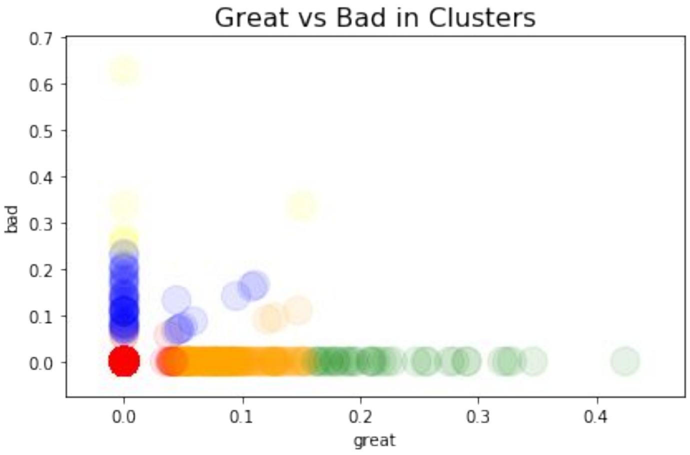
K-means of selected features (words)

```
[52] kmModelSelected = KMeans(n_clusters = k)
     kmModelSelected.fit(tfidf_reviewDF[['great', 'bad']])
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

```
[53] tfidf_reviewDF['cluster'] = kmModelSelected.labels_
```

```
[54] # Create the color palette:
     colorPalette = { 0: 'red', 1: 'green', 2: 'blue', 3: 'yellow', 4: 'orange' }
     colors = tfidf_reviewDF.apply(lambda row: colorPalette[row.cluster], axis = 1)

     # Create a scatter plot of "great" vs "bad" in clusters:
     tfidf_reviewDF.plot(kind = 'scatter', x = 'great', y = 'bad', alpha = 0.1, s = 300, c = colors)

     plt.xlabel("great")
     plt.ylabel("bad")

     plt.title("Great vs Bad in Clusters", fontsize = 16)

     plt.tight_layout()
```



Lu Liu & Daniel Amin. (2019)

Logistic Regression: Implementation & Result

```
[ ] logRegModel = LogisticRegression()

    logRegModel.fit(reviewArray_train, labelArray_train)
```

```
[ ] label_predict_logReg = logRegModel.predict(reviewArray_test)
```

```
[ ] label_predict_logReg
```

```
array([1, 1, 2, 1, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2,
       1, 2, 2, 1, 2, 1, 1, 2, 2, 2, 1, 2, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1,
       2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 1, 2, 1, 2, 1, 2, 1, 1, 1, 2, 2,
       2, 2, 1, 1, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 1, 2, 2, 1, 1, 1, 2,
       1, 2, 2, 2, 1, 2, 2, 1, 2, 1, 2, 1, 1, 2, 2, 2, 2, 1, 1, 1, 2, 1,
       2, 1, 2, 1, 1, 2, 1, 2, 2, 2, 2, 1, 1, 2, 2, 1, 2, 1, 1, 2, 1, 1,
       1, 1, 2, 2, 1, 1, 2, 1, 2, 2, 1, 2, 2, 2, 1, 2, 1, 2, 2, 2, 1, 1,
       2, 1, 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 1, 1, 2, 2, 2, 2,
       1, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 1,
       1, 2])
```

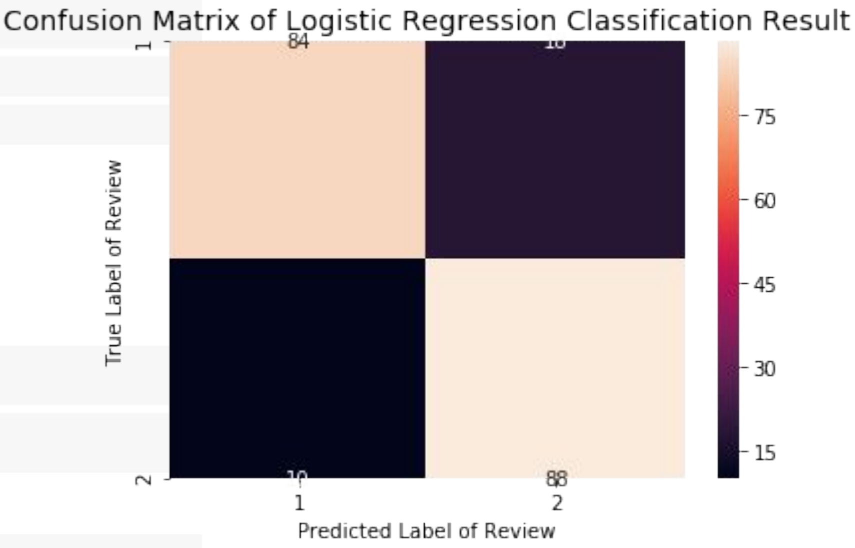
```
[ ] # Import confusion_matrix from Scikit Learn for accuracy evaluation:
    from sklearn.metrics import confusion_matrix
```

```
[ ] confMetrix_logReg = confusion_matrix(labelArray_test, label_predict_logReg)
    confMetrix_logReg
```

```
array([[84, 18],
       [10, 88]])
```

```
[ ] # Import accuracy_score from Scikit Learn for computing the prediction accuracy:
    from sklearn.metrics import accuracy_score
```

```
[ ] print("Accuracy for Logistic Regression Classifier: " + str(accuracy_score(labelArray_test, label_predict_logReg) * 100) + "%")
```



Accuracy for Logistic Regression Classifier: 86.0%

Lu Liu & Daniel Amin. (2019)

SVM Classification: Implementation & Result

```
[ ] # Import SVM from Sciket Learn for classification:
    from sklearn import svm
```

```
[ ] svmModel = svm.SVC(kernel = 'linear')

    svmModel.fit(reviewArray_train, labelArray_train)
```

```
[ ] label_predict_svm = svmModel.predict(reviewArray_test)
```

```
[ ] label_predict_svm
```

```
array([1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2,
       1, 2, 2, 1, 2, 1, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 1, 2, 1, 1, 1,
       2, 2, 2, 2, 2, 2, 1, 2, 1, 1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 1, 2, 2,
       1, 2, 1, 1, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 1, 2, 2, 1, 1, 1, 2,
       1, 2, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 1, 2, 1, 2, 2, 2, 1, 1, 2, 1,
       2, 1, 2, 1, 1, 2, 1, 2, 2, 2, 2, 1, 1, 2, 2, 1, 2, 1, 1, 2, 1, 1,
       1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 1, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 1,
       1, 1, 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 2, 1, 2, 2, 1, 1, 2, 2, 2, 2,
       1, 2, 1, 1, 2, 1, 1, 2, 1, 2, 2, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 1,
       1, 2])
```

```
[ ] # Import confusion_matrix from Scikit Learn for accuracy evaluation:
    from sklearn.metrics import confusion_matrix
```

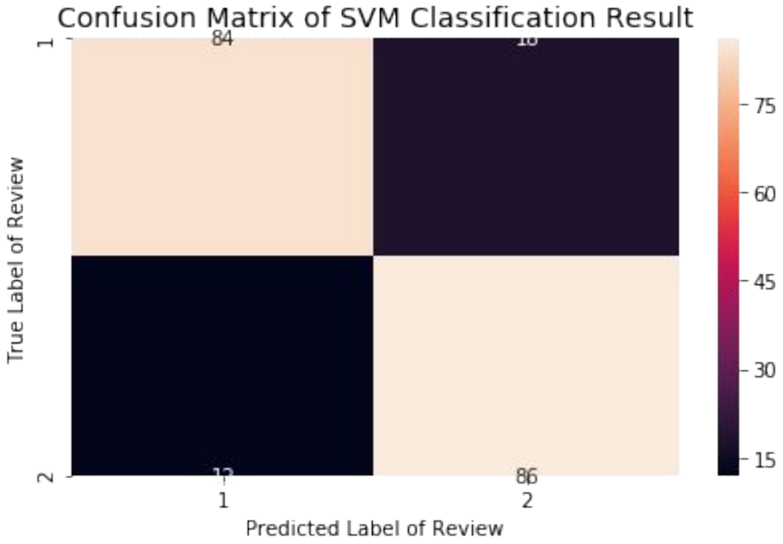
```
[ ] confMetrix_svm = confusion_matrix(labelArray_test, label_predict_svm)
    confMetrix_svm
```

```
array([[84, 18],
       [12, 86]])
```

```
[ ] # Import accuracy_score from Scikit Learn for computing the prediction accuracy:
    from sklearn.metrics import accuracy_score
```

```
[ ] print("Accuracy for SVM Classifier: " + str(accuracy_score(labelArray_test, label_predict_svm) * 100) + "%")
```

Accuracy for SVM Classifier: 85.0%



Lu Liu & Daniel Amin. (2019)

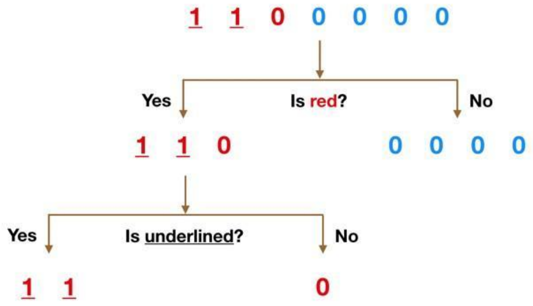
Algorithm: Random Forest

Random Forest “consists of a large number of individual decision trees that operate as an *ensemble*. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction ” (*Yiu, towardsdatascience.com*).

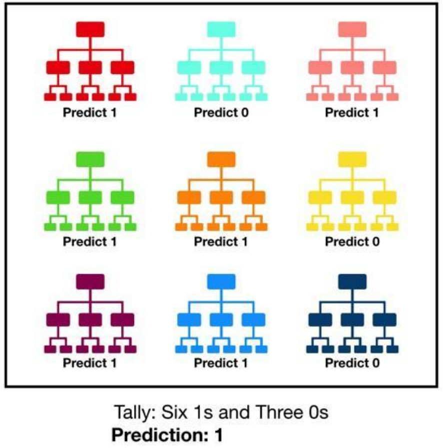
Remarks: Random Forest *outperforms* any of the individual constituent models - Does not overfitting

- Low correlation between models
- The trees protect each other from their individual errors

Package(s): sklearn.ensemble.RandomForestClassifier



Decision Tree (Image via Yiu)

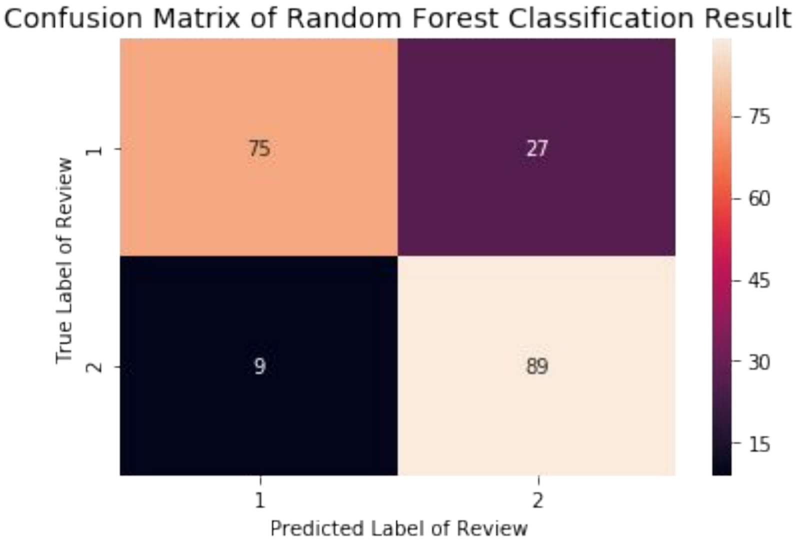


Random Forest (Image via Yiu)

References:
https://towardsdatascience.com/understanding-random-forest-58381e0602d2
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Random Forest Classification: Implementation

```
[51] from sklearn.ensemble import RandomForestClassifier

[52] # Set the parameter random_state with an abitary number so that the same seed is used by the random number generator in every run.
     rfModel = RandomForestClassifier(n_estimators = 500, criterion = 'entropy', random_state = 456)

     rfModel.fit(reviewArray_train, labelArray_train)

     RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                            max_depth=None, max_features='auto', max_leaf_nodes=None,
                            min_impurity_decrease=0.0, min_impurity_split=None,
                            min_samples_leaf=1, min_samples_split=2,
                            min_weight_fraction_leaf=0.0, n_estimators=500,
                            n_jobs=None, oob_score=False, random_state=456,
                            verbose=0, warm_start=False)

[53] label_predict_rf = rfModel.predict(reviewArray_test)

[54] label_predict_rf

     array([1, 2, 2, 1, 2, 2, 2, 2, 1, 1, 2, 2, 2, 1, 2, 1, 1, 1, 2, 2, 2, 2,
            2, 2, 2, 1, 2, 1, 1, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 2, 1, 1,
            2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 1, 1, 1, 2, 2,
            1, 2, 2, 1, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 1, 2, 2, 1, 1, 1, 2,
            1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 2, 1, 1, 2, 2, 1, 2, 2, 1, 1, 2, 1,
            2, 1, 1, 2, 1, 2, 1, 2, 2, 2, 2, 1, 1, 2, 2, 1, 2, 1, 1, 2, 1, 1,
            1, 1, 2, 2, 1, 1, 2, 1, 2, 2, 1, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 2,
            2, 1, 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2,
            1, 2, 1, 1, 2, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 1, 1, 1, 1, 2, 2, 1,
            1, 2])

[55] # Import confusion_matrix from Scikit Learn for accuracy evaluation:
     from sklearn.metrics import confusion_matrix

[56] confMetrix_rf = confusion_matrix(labelArray_test, label_predict_rf)
     confMetrix_rf

     array([[75, 27],
            [ 9, 89]])
```



Accuracy for Random Forest Classifier: 82.0% (n_estimators = 500)

→ Random Forest may not be very ideal for high-dimensional sparse data, e.g., Bag-of-Word, given only 1,000 reviews (training 80%, testing 20%) are used as a sub-dataset in this Random Forest experiment.

Lu Liu & Daniel Amin. (2019)

Improved Approaches

Training Data: 120K Reviews, Testing Data: 40K Reviews

Labels: “0” for Negative Reviews, “1” for Positive Reviews

- Data Resampling & Repreprocessing
- Binary Cross-Entropy Loss
- Word2Vec
- Deep Neural Net
- CNN
- Combined Random Forest Classifier
- Results from all model reruns
- VADER Lexicon

Improved Approach: Binary Cross-Entropy Loss

Loss Functions return “high values for bad predictions and low values for good predictions” (*Godoy, towardsdatascience.com*)

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Binary Cross-Entropy / Log Loss

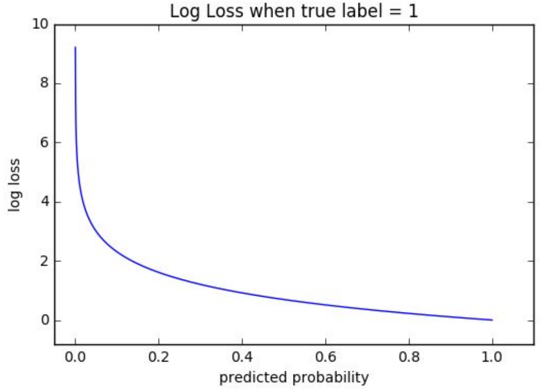
(Image via Godoy)

Binary Cross-Entropy / Log Loss is the typical loss function for binary classification.

Remarks:

- In order to improve the performances of our models, we changed the labels of reviews from “1” (negative) and “2” (positive) to “0” (negative) and “1” (positive) and retrained the models. The change actually generated almost 20% of accuracy with Deep Neural Net and CNN.

Package(s): TensorFlow



(Image via Machine Learning Glossary)

References:
https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a
https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

Data Preprocessing

Data Resampling:

- Training Data: 120K Reviews
- Testing Data: 40K Reviews

```
[ ] dataTrain = pd.DataFrame()
    dataTrain['text'] = lines
    dataTrain['label'] = labels

    dataTrain = dataTrain.sample(n = 120000, random_state = 123)
    print(len(dataTrain))
    dataTrain.head()
```

```
120000
```

	text	label
2725661	this movie sucks: This movie supposedly about ...	0
1798719	Good Entertainment: This program a well edited...	0
1242154	Does the job: This hamper does the job in my k...	1
3373098	Buffett Mails it In: Being a huge Buffett fan,...	0
1663895	Sharp as a razor... almost.: Wow! My replaceme...	1

```
[ ] dataTest = pd.DataFrame()
    dataTest['text'] = lines
    dataTest['label'] = labels

    dataTest = dataTest.sample(n = 40000, random_state = 456)
    print(len(dataTest))
    dataTest.head()
```

```
40000
```

	text	label
333305	Confused: I have been a science fiction/fantas...	0
27936	What a SORRY A$$ way to go out!: Since this is...	0
17999	If I had my way, I'd have all of you shot: I I...	1
124332	Super Fun for My Super Heroes!: You cannot eve...	1
303110	Extremely Poor Quality: This bit set is absolu...	0

Lu Liu & Daniel Amin. (2019)

Data Preprocessing

Replacing Labels:

- "**__label__1**": using integer **0** for Negative Review
- "**__label__2**": using integer **1** for Positive Review

Changing to lowercase & removing punctuations, stopwords

Main function for cleaning texts:

```
[ ] stop = stopwords.words('english')
    w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
    lemmatizer = nltk.stem.WordNetLemmatizer()

    # Define the function for lemmatizing texts:
    def lemmatize_text(text):
        return [lemmatizer.lemmatize(w) for w in w_tokenizer.tokenize(text)]

    # Define the function for changing to lowercase, removing punctuation and stopwords:
    def clean_text(text):
        text = text.str.lower()
        text = text.str.replace('-', ' ')
        text = text.str.split(' ')
        text = text.apply(lambda x: [item for item in x if item not in stop])
        text = text.apply(', '.join)
        text = text.str.replace('[{}]'.format(string.punctuation), '')
        text = text.apply(lemmatize_text)
        text = text.apply(', '.join)
        text = text.str.replace('[{}]'.format(string.punctuation), '')
        text = text.str.replace('\\', ' ')
        return text
```

```
[ ] # Define the function for replacing label strings with integer 0 or 1:
    def replace_label(text):
        labels = []

        for item in text:
            first_ten_chars = item[:10]
            if first_ten_chars == '__label__1':
                labels.append(int(0))    # 0 for Negative Review: '__label__1'
            elif first_ten_chars == '__label__2':
                labels.append(int(1))    # 1 for Positive Review: '__label__2'

        return labels

    labels = replace_label(lines)
```

```
[ ] # Define the function for removing lable strings in lines:
    def remove_label(s):
        return s[11:]    # The text review starts from index 11 to the last index.

    lines = [remove_label(s) for s in lines]
```

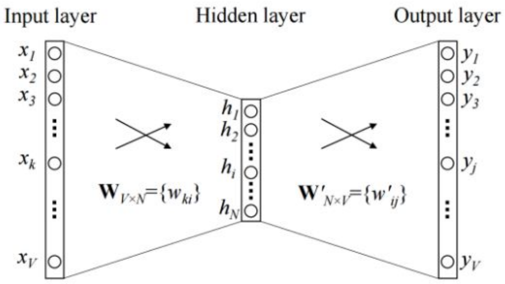

Lu Liu & Daniel Amin. (2019)

Improved Approach: Word2Vec

Word2Vec: Efficient word embedding algorithm using neural network that embeds word as a vector of size 100-300. Works by using corpus of text to find statistical knowledge of word occurences, in which each word is mapped to a certain space by its similarities with other words in terms of occurrence (Mikolov, Chen, Corrado, & Dean, 2013).

CBOW (Common Bag of Words): Input context words to predict target word.

- Use the one hot encoding of the input word and measure the output error compared to one hot encoding of the target word, in the process learn vector representation of the word (Karani, towardsdatascience.com).



Cbow W2V (Image via Karani)

References:
https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa
Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. 1–12. Retrieved from http://arxiv.org/abs/1301.3781

Word2Vec & BOW

```
model_w2v.wv.most_similar('good')
```

```
[('decent', 0.7589436769485474),
 ('great', 0.7339096069335938),
 ('bad', 0.6841711401939392),
 ('ok', 0.6451138257980347),
 ('okay', 0.632263720035553),
 ('excellent', 0.6051275730133057),
 ('nice', 0.6029865145683289),
 ('fair', 0.5386767983436584),
 ('cool', 0.5366443395614624),
 ('neat', 0.5114398002624512)]
```

Package(s): Gensim

Each word in the row is represented by 100-dim vector, then for every row, these vectors are averaged, implementing BOW.

```
model_w2v['good']
```

```
/home/danielamin/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py
self.wv.__getitem__() instead).
  """Entry point for launching an IPython kernel.
```

```
array([ 2.1755898 ,  0.10826331,  0.12476164,  1.16604   , -1.4839977 ,
       -0.34801066,  0.35136965,  0.6432537 ,  0.18190865,  0.9152566 ,
        1.0175338 , -2.1535182 , -0.67739576, -0.55686826, -1.4887441 ,
        0.25776526,  0.6663208 , -1.3809274 , -0.778668  ,  1.1670469 ,
       -0.9147078 , -1.7038858 ,  0.25083682,  1.4087104 , -0.17531088,
        0.0738984 ,  0.1594836 , -1.6888974 , -2.8572688 ,  1.4356275 ,
       -0.58971053,  1.187398  ,  2.1166122 ,  1.716     ,  1.7438118 ,
        0.02434559, -0.04338304, -0.64896834, -0.9149263 , -1.875239  ,
       -0.6007584 ,  1.2118523 ,  0.970247  ,  1.5339544 ,  0.5500239 ,
        1.0200495 , -0.8799366 ,  0.10331298,  0.62382936,  2.1578434 ,
        0.59559375, -1.0809524 ,  0.26148614, -1.3784628 ,  0.34488118,
        1.1577228 , -1.5415885 ,  1.9827507 ,  1.7880238 ,  0.13731083,
       -0.25919384, -2.221568  , -0.34264836,  2.3132684 ,  1.3040881 ,
       -0.22553805,  1.0905842 ,  0.22230984, -0.6318164 , -1.3535357 ,
       -0.2919598 ,  0.4751481 , -1.1512574 , -0.33107588, -0.607018  ,
       -0.262131  , -0.53740174,  1.2289382 , -0.8431087 ,  1.5677354 ,
        0.48949894, -0.13956273,  0.9554281 , -0.8327267 , -0.5382966 ,
        0.687798  , -0.24940318, -0.4823991 , -1.0253251 ,  1.3505661 ,
       -0.5641778 , -0.42182967, -0.05678038, -2.1001656 , -2.0510702 ,
        1.1768535 ,  0.5502338 ,  0.46044078, -0.20649819,  0.06546904],
      dtype=float32)
```


Lu Liu & Daniel Amin. (2019)

Deep Neural Net: Implementation and Result

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, load_model

# mini batches Nadam optimizer with dropout and batch normalization
epochs = 100
model = tf.keras.Sequential()
model.add(layers.Dense(32, input_dim=100))
model.add(layers.Activation('relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dense(32))
model.add(layers.Activation('relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(rate = 0.2))
model.add(layers.Dense(32))
model.add(layers.Activation('relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dense(64))
model.add(layers.Activation('relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(rate = 0.3))
model.add(layers.Dense(64))
model.add(layers.Activation('relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dense(64))
model.add(layers.Activation('relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(rate = 0.4))

model.add(layers.Dense(1))
model.add(layers.Activation('sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer=keras.optimizers.Nadam(lr=0.002, beta_1=0.9, beta_2=0.999),
              metrics=['accuracy'])
checkpoint = keras.callbacks.ModelCheckpoint("NN.model", monitor='val_accuracy', verbose=1, save_best_only=True)

model.summary()
model1 = model.fit(x_train, y_train, epochs=epochs, validation_split=0.2, callbacks=[checkpoint])
#history = model.fit(x_train, y_train, epochs = epochs, validation_split=0.2)
```

Credits: Emily Campbell & Stephen Riesenberg

Ran 100 epochs: 86.2% (on best epoch)

→ Trained 100-dim Word2Vec for efficient word embedding, used binary cross-entropy loss

CNN: Implementation and Result

```python
import tensorflow as tf

shape = (x_train.shape[1], 1)
model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Conv1D(32, kernel_size=3, activation=tf.nn.relu, input_shape=shape))
model.add(tf.keras.layers.Conv1D(32, kernel_size=3, activation=tf.nn.relu))

model.add(tf.keras.layers.MaxPooling1D(pool_size=2))

model.add(tf.keras.layers.Conv1D(64, kernel_size=3, activation=tf.nn.relu))
model.add(tf.keras.layers.Conv1D(64, kernel_size=3, activation=tf.nn.relu))

model.add(tf.keras.layers.MaxPooling1D(pool_size=2))

model.add(tf.keras.layers.Conv1D(128, kernel_size=3, activation=tf.nn.relu))
model.add(tf.keras.layers.Conv1D(128, kernel_size=3, activation=tf.nn.relu))

model.add(tf.keras.layers.MaxPooling1D(pool_size=2))

model.add(tf.keras.layers.Conv1D(256, kernel_size=3, activation=tf.nn.relu))
model.add(tf.keras.layers.Conv1D(256, kernel_size=3, activation=tf.nn.relu))

model.add(tf.keras.layers.MaxPooling1D(pool_size=5))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.BatchNormalization())

model.add(tf.keras.layers.Dense(200, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(100, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(1, activation=tf.nn.sigmoid))


model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Conneau, A., Schwenk, H., Barrault, L., & Lecun, Y. (2017). *Very Deep Convolutional Networks for Text Classification*. Retrieved from https://arxiv.org/abs/1606.01781

Ran 100 epochs: 85.4% (on best epoch)

→ Trained 100-dim Word2Vec for efficient word embedding, used binary cross-entropy loss

Combined Random Forest Classifier

Algorithm Structure: From algorithms such as Logistic Regression, Bayes, SVM, Random Forest, NN, and CNN with the Word2Vec BOW features, the output of these algorithms are used as input for a second layer of classification using Random Forest.

```
lr_predict = lr.predict(x_train)
nb_predict = clf.predict(x_train)
svm_predict = svm_model.predict(x_train)
rf_predict = rf.predict(x_train)
nn_predict = nn_model.predict_classes(x_train)
cnn_predict = cnn_model.predict_classes(x_train1)
```

```
new_features = pd.DataFrame()
new_features['lr_predict'] = lr_predict
new_features['nb_predict'] = nb_predict
new_features['svm_predict'] = svm_predict
new_features['rf_predict'] = rf_predict
new_features['nn_predict'] = nn_predict
new_features['cnn_predict'] = cnn_predict

new_features.head()
```

	lr_predict	nb_predict	svm_predict	rf_predict	nn_predict	cnn_predict
0	0	0	0	0	0	0
1	1	1	1	1	1	1
2	1	0	1	0	1	1
3	1	1	1	1	1	1
4	1	0	1	1	1	1

RandomForestClassifier(n_estimators=500, criterion='entropy')

Accuracy: 83.2%

Package(s): Scikit-learn

Lu Liu & Daniel Amin. (2019)

Results with Word2Vec

Logistic Regression

Label	Acc	Prec	Rec	F1	Supp
0	0.850	0.843	0.858	0.851	19996
1		0.856	0.841	0.848	20004

Gaussian Bayes

Label	Acc	Prec	Rec	F1	Supp
0	0.725	0.728	0.720	0.724	19996
1		0.723	0.731	0.727	20004

Linear-kernel SVM

Label	Acc	Prec	Rec	F1	Supp
0	0.849	0.844	0.857	0.851	19996
1		0.855	0.842	0.848	20004

Random Forest

Label	Acc	Prec	Rec	F1	Supp
0	0.832	0.825	0.843	0.834	19996
1		0.840	0.821	0.830	20004

Results with Word2Vec

Deep Neural Net

Label	Acc	Prec	Rec	F1	Supp
0	0.862	0.863	0.860	0.861	19996
1		0.861	0.863	0.862	20004

CNN

Label	Acc	Prec	Rec	F1	Supp
0	0.854	0.853	0.855	0.854	19996
1		0.855	0.853	0.854	20004

Combined(All models as input to a RF Classifier)

Label	Acc	Prec	Rec	F1	Supp
0	0.832	0.825	0.843	0.834	19996
1		0.840	0.821	0.830	20004

Emotion Lexicon for Sentiment Analysis

Next Steps:

- Find the intersection between the bag-of-words results with the emotion lexicon.
- Challenges:
  - Some words may be associated with multiple emotions
  - Each document could be associated with multiple emotions
- Solution:
  - Label the emotion of each document (each block/paragraph of customer comment) based on the distribution of each emotion.
- Some lexicon resources (Sarkar. 2018):
  - AFINN lexicon
  - Bing Liu's lexicon
  - MPQA subjectivity lexicon
  - SentiWordNet
  - VADER lexicon
  - TextBlob lexicon

VADER Lexicon

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is *specifically attuned to sentiments expressed in social media*, and works well on texts from other domains (*Hutto, C.J. & Gilbert, E.E. 2014*).

Remarks:

- Use of punctuations (e.g., "Good!!!")
- Understanding many sentiment-laden slang words (e.g., 'uber' or 'friggin' or 'kinda')
- Translating utf-8 encoded emojis such as 💘 and 💋 and 😁
- Understanding sentiment-laden initialisms and acronyms (for example: 'lol')

Package(s): nltk.sentiment.vader

References:
Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.
https://www.nltk.org/_modules/nltk/sentiment/vader.html

VADER Lexicon

Sentiment Analysis

```
[ ] # Import SentimentIntensityAnalyzer from NLTK Vader lexicon:
    from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

```
[ ] # Define the Vader sentiment analyzer:
    vaderAnalyzer = SentimentIntensityAnalyzer()

    # Define the function of review rate:
    def sentiment_analyzer_scores(text):
      score = vaderAnalyzer.polarity_scores(text)
      print(text)
      print(score)
      print()

      print("Overall, the review is rated as: ")

      if score['compound'] <= -0.05:
        print("Negative (Label: 1)")

      elif score['compound'] >= 0.05:
        print("Positive (Label: 2)")

      else:
        print("Neutral (No Label)")

      print()
      print("-------------------------------------")
      print()
```

```
[ ] text1 = "Lmao, it's so cute~"
    text2 = "im very disappointed"
    text3 = "The price is good but the quality is trash!"

    sentiment_analyzer_scores(text1)
    sentiment_analyzer_scores(text2)
    sentiment_analyzer_scores(text3)
```

```
Lmao, it's so cute~
{'neg': 0.0, 'neu': 0.435, 'pos': 0.565, 'compound': 0.5994}

Overall, the review is rated as:
Positive (Label: 2)

-------------------------------------

im very disappointed
{'neg': 0.629, 'neu': 0.371, 'pos': 0.0, 'compound': -0.5256}

Overall, the review is rated as:
Negative (Label: 1)

-------------------------------------

The price is good but the quality is trash!
{'neg': 0.0, 'neu': 0.781, 'pos': 0.219, 'compound': 0.3054}

Overall, the review is rated as:
Positive (Label: 2)

-------------------------------------
```

```
[ ] print("=> Original Review:")
    sentiment_analyzer_scores(df1.iloc[500, 1])
    # print(df1.iloc[0, 1])
    # print("=> Sentiment Analysis of the Original Review:")
    # print(vaderAnalyzer.polarity_scores(df1.iloc[0, 1]))
    # print()

    print("=> Cleaned Text of the review:")
    sentiment_analyzer_scores(reviewCorpus[500])
    # print(reviewCorpus[0])
    # print("=> Sentiment Analysis of the Cleaned Text:")
    # print(vaderAnalyzer.polarity_scores(reviewCorpus[0]))

    print("Actual Label: ",labelArray[500])
```

```
=> Original Review:
misleading photo: There was little information about the product other than the photo
{'neg': 0.047, 'neu': 0.829, 'pos': 0.124, 'compound': 0.7707}

Overall, the review is rated as:
Positive (Label: 2)

-------------------------------------

=> Cleaned Text of the review:
misleading photo little information product photo showed watchtower toy action figure
{'neg': 0.145, 'neu': 0.67, 'pos': 0.186, 'compound': 0.4404}

Overall, the review is rated as:
Positive (Label: 2)

-------------------------------------

Actual Label:  1
```

References: Riso, R. Sentiment Analysis: Beyond Words. Retrieved from https://towardsdatascience.com/sentiment-analysis-beyond-words-6ca17a6c1b54

VADER Lexicon

```
[ ] vaderAnalyzer = SentimentIntensityAnalyzer()

    # Define the function for analyzing sentiment using Vader Lexicon:
    def analyze_sentiment_label(text):
        sentimentAnalysisResult = []

        for i in range(0, len(text)):
            score = vaderAnalyzer.polarity_scores(text[i])

            if score['compound'] <= -0.05:
                sentimentAnalysisResult.append(int(0))    # Negative Reviews
            elif score['compound'] >= 0.05:
                sentimentAnalysisResult.append(int(1))    # Positive Reviews
            else:
                sentimentAnalysisResult.append(int(2))    # Neutral Reviews (Do not exist in the original dataset)

        return sentimentAnalysisResult

[ ] len(dataTrain)

120000

[ ] # Covert the panda.series to numpy array datatype with ".values":
    reviewTrain = dataTrain['text'].values

    print(len(reviewTrain))
    print(type(reviewTrain))

120000
<class 'numpy.ndarray'>

[ ] # Get Vader analysis results:
    vaderLabels_train = analyze_sentiment_label(reviewTrain)

[ ] print(type(vaderLabels_train))

<class 'list'>

[ ] print("Confusion Matrix:")
    print(confusion_matrix(dataTrain['label'].tolist(), vaderLabels))

Confusion Matrix:
[[23596 34030  2483]
 [ 3238 55846   807]
 [    0     0     0]]

[ ] print("Accuracy on Training Data: " + str(accuracy_score(dataTrain['label'].tolist(), vaderLabels_train) * 100) + "%")

Accuracy on Training Data: 66.20166666666667%
```

VADER contains score range for neutral sentiment, which is coded here as "2" but do not exist in the original dataset.

```
[ ] len(dataTest)

40000

[ ] # Covert the panda.series to numpy array datatype with ".values":
    reviewTest = dataTest['text'].values

    print(len(reviewTest))
    print(type(reviewTest))

40000
<class 'numpy.ndarray'>

[ ] # Get Vader analysis results:
    vaderLabels_test = analyze_sentiment_label(reviewTest)

[ ] print(type(vaderLabels_test))

<class 'list'>

[ ] print("Confusion Matrix:")
    print(confusion_matrix(dataTest['label'].tolist(), vaderLabels_test))

Confusion Matrix:
[[ 7911 11438   771]
 [ 1108 18505   267]
 [    0     0     0]]

[ ] print("Accuracy on Testing Data: " + str(accuracy_score(dataTest['label'].tolist(), vaderLabels_test) * 100) + "%")

Accuracy on Testing Data: 66.03999999999999%
```

Accuracy:

- Training Data: 66.2%
- Testing Data: 66.0%

Lu Liu & Daniel Amin. (2019)

Conclusion on Classification Algorithms

Algorithm	*Naive Bayes*	*Logistic Regression*	*SVM*	*Random Forest*	*Deep Neural Net*	*CNN*	*Combined Random Forest*	*VADER Lexicon*
Type	Classification	Classification	Classification	Classification	Classification	Classification	Classification	Classification
Accuracy	72.5%	85.0%	84.9%	83.2%	86.2%	85.4%	83.2%	66.2%

- In the comparison table, **Deep Neural Net** achieved *the highest accuracy* while **CNN** and **Logistic Regression** came right after. The performance of the **Naive Bayes classifier** had *the lowest accuracy* among all algorithms that used the same dataset of labels.
- Surprisingly, *there wasn't any improvement* on the performance of our **Combined Random Forest classifier** -- The 2nd layer failed to capture any more details than the RF on the first layer. **Deep Neural Net** with original vectors still won!
- *The original dataset doesn't include any neutral review label*. That may lead to the low accuracy of the sentiment analysis via **VADER Lexicon**.

Lu Liu & Daniel Amin. (2019)

Future Work: Emotion Lexicon

Another Proposed Approach: **Word-Emotion Association** (a.k.a. **NRC Emotion Lexicon**) from National Research Council Canada (*Source: http://sentiment.nrc.ca/lexicons-for-research/*)

- 8 Emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and 2 Sentiments (negative and positive)
- Number of Terms:
  - >14,000 unigrams (words)
  - ~25,000 word senses
- Association scores: Binary (associated or not)

aback	anger	0
aback	anticipation	0
aback	disgust	0
aback	fear	0
aback	joy	0
aback	negative	0
aback	positive	0
aback	sadness	0
aback	surprise	0
aback	trust	0
abacus	anger	0
abacus	anticipation	0
abacus	disgust	0
abacus	fear	0
abacus	joy	0
abacus	negative	0
abacus	positive	0
abacus	sadness	0
abacus	surprise	0
abacus	trust	1

Reason:

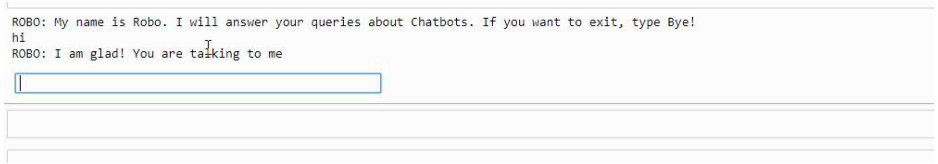
- This lexicon is more recognized and inclusive compared to the Emotion Sensor Dataset from Kaggle
- It contains both emotion and sentiment corpuses

Future Work: Chatbot

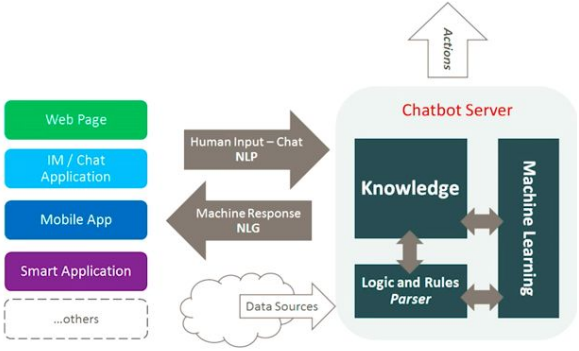
Phase 3: Incorporating Classifier with Text Search in a Retrieval-Based Chatbot

- "In retrieval-based models, a chatbot uses some heuristic to select a response from a library of predefined responses. The chatbot uses the message and context of the conversation for selecting the best response from a predefined list of bot messages" (Pandey, 2018).

Package(s): Scikit Learn, NLTK



Anatomy of a Chatbot



References:
Pandey, P. 2018. Building a Simple Chatbot from Scratch in Python (using NLTK). Retrieved from [*https://medium.com/analytics-vidhya/building-a-simple-chatbot-in-python-using-nltk-7c8c8215ac6e*](https://medium.com/analytics-vidhya/building-a-simple-chatbot-in-python-using-nltk-7c8c8215ac6e)

References

- Agents that Learn - UWA. Retrieved from http://teaching.csse.uwa.edu.au/units/CITS4211/Lectures/wk5.pdf
- Brownlee, J. (2015). Basic Concepts in Machine Learning. Machine Learning Mastery. Retrieved from https://machinelearningmastery.com/basic-concepts-in-machine-learning/
- Cardie, C. (2012). CS4740/CS5740/LING4474/CGSCI4740 Introduction To Natural Language Processsing. Cornell University. Retrieved December 15, 2019, from https://www.cs.cornell.edu/courses/cs4740/2012sp/
- Sarkar. (2018). Emotion and Sentiment Analysis: A Practitioner's Guide to NLP. Retrieved from https://www.kdnuggets.com/2018/08/emotion-sentiment-analysis-practitioners-guide-nlp-5.html
- An introduction to Bag of Words and how to code it in Python for NLP. Retrieved from https://www.freecodecamp.org/news/an-introduction-to-bag-of-words-and-how-to-code-it-in-python-for-nlp-282e87a9da04/
- Python for NLP: Creating Bag of Words Model from Scratch. Retrieved from https://stackabuse.com/python-for-nlp-creating-bag-of-words-model-from-scratch/
- 5.2. Feature extraction — scikit-learn 0.21.3 documentation. Retrieved from https://scikit-learn.org/stable/modules/feature_extraction.html
- A Gentle Introduction to the Bag-of-Words Model. Retrieved from https://machinelearningmastery.com/gentle-introduction-bag-words-model/
- Python | NLP analysis of Restaurant reviews. GeeksforGeeks.org. Retrieved from https://www.geeksforgeeks.org/python-nlp-analysis-of-restaurant-reviews/
- Compute Precision Call Accuracy Sklearn. Retrieved from https://stackoverflow.com/questions/31421413/how-to-compute-precision-recall-accuracy-and-f1-score-for-the-multiclass-case

References (Cont.)

- sklearn.feature_extraction.text.TfidfVectorizer — scikit-learn 0.21.3 documentation. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
- K-Means Clustering with scikit-learn. Retrieved from http://jonathansoma.com/lede/algorithms-2017/classes/clustering/k-means-clustering-with-scikit-learn/
- kmeans text clustering | Python Tutorial. Retrieved from https://pythonprogramminglanguage.com/kmeans-text-clustering/
- In Depth: k-Means Clustering | Python Data Science Handbook. Retrieved from https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html
- Salnikov, M. Text clustering with K-means and tf-idf - Mikhail Salnikov - Medium. Retrieved from https://medium.com/@MSalnikov/text-clustering-with-k-means-and-tf-idf-f099bcf95183
- Conneau, A., Schwenk, H., Barrault, L., & Lecun, Y. (2017). *Very Deep Convolutional Networks for Text Classification*. Retrieved from https://arxiv.org/abs/1606.01781
- Li, S. Scikit-Learn for Text Analysis of Amazon Fine Food Reviews. datascience+. Retrieved from https://datascienceplus.com/scikit-learn-for-text-analysis-of-amazon-fine-food-reviews/
- Yiu, T. Understanding Random Forest - Towards Data Science. Retrieved from https://towardsdatascience.com/understanding-random-forest-58381e0602d2
- Godoy, D. (2018, November 21). Understanding Binary Cross-entropy / Log Loss: A Visual Explanation. Medium. Retrieved December 15, 2019, from https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a
- (n.d.). Loss Functions — ML Glossary Documentation. Machine Learning Glossary. Retrieved December 15, 2019, from https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

References (Cont.)

- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. 1–12. Retrieved from http://arxiv.org/abs/1301.3781
- Sarkar, D. RedHat. Emotion and Sentiment Analysis: A Practitioner's Guide to NLP. Retrieved from https://www.kdnuggets.com/2018/08/emotion-sentiment-analysis-practitioners-guide-nlp-5.html
- Python | Sentiment Analysis using VADER. GeeksforGeeks. Retrieved from https://www.geeksforgeeks.org/python-sentiment-analysis-using-vader/
- Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.
- Hutto, C.J. cjhutto/vaderSentiment. GitHub. Retrieved from https://github.com/cjhutto/vaderSentiment
- Rafferty, G. Sentiment Analysis on the Texts of Harry Potter. Retrieved from https://towardsdatascience.com/basic-nlp-on-the-texts-of-harry-potter-sentiment-analysis-1b474b13651d
- Rafferty, G. raffg/harry_potter_nlp. GitHub. Retrieved from https://github.com/raffg/harry_potter_nlp/blob/master/sentiment_analysis.ipynb
- Riso, R. Sentiment Analysis: Beyond Words. Retrieved from https://towardsdatascience.com/sentiment-analysis-beyond-words-6ca17a6c1b54
- Soma, J. NRC Emotional Lexicon. Retrieved from http://jonathansoma.com/lede/algorithms-2017/classes/more-text-analysis/nrc-emotional-lexicon/
- Pandey, P. 2018. Building a Simple Chatbot from Scratch in Python (using NLTK). Retrieved from https://medium.com/analytics-vidhya/building-a-simple-chatbot-in-python-using-nltk-7c8c8215ac6e

Q & A

Thank You