Intro. to Natural Language Processing

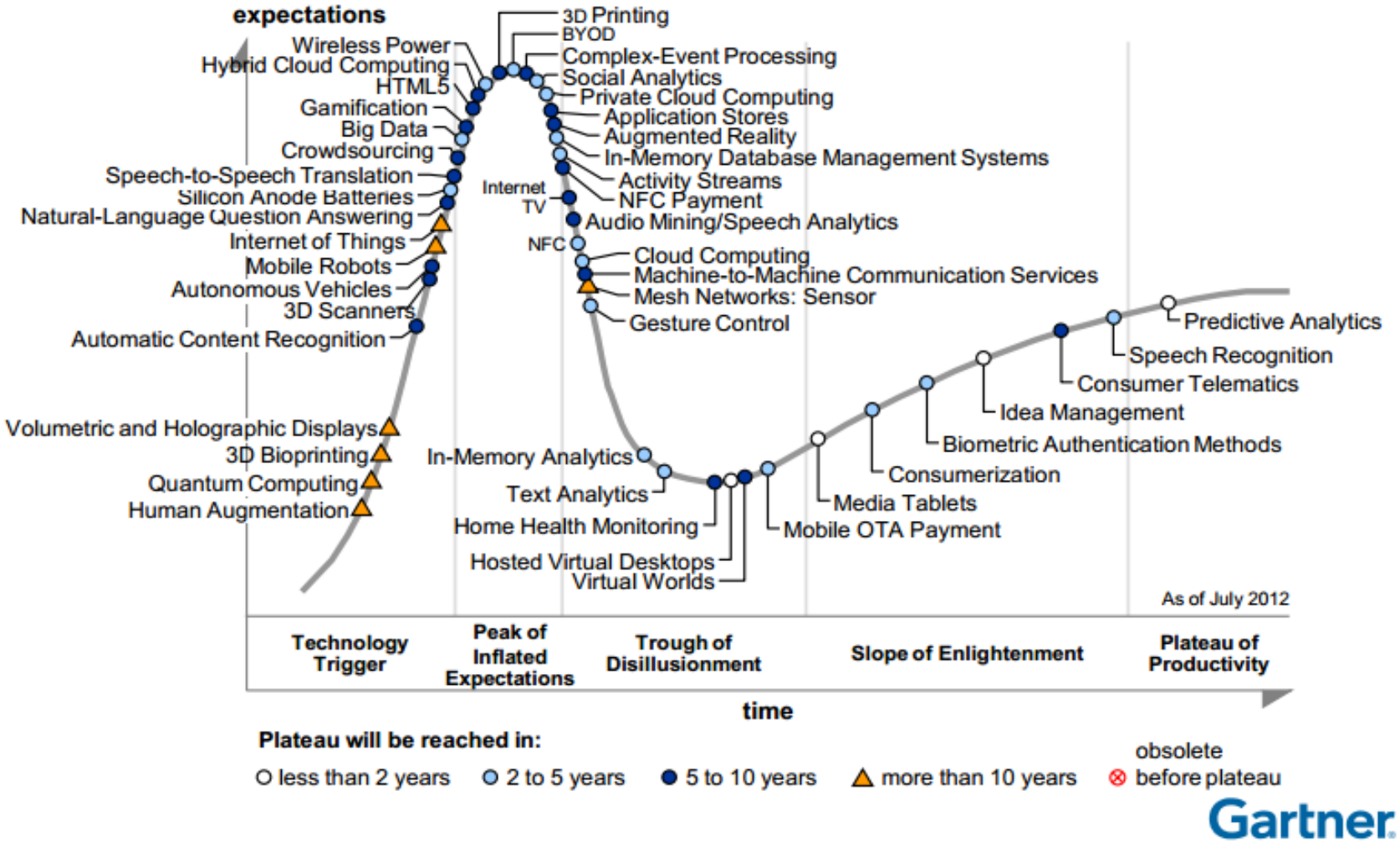
Pree Thiengburanathum, PhD.

Develop Data Science Machine learning for Business
14th October 2021

Agenda

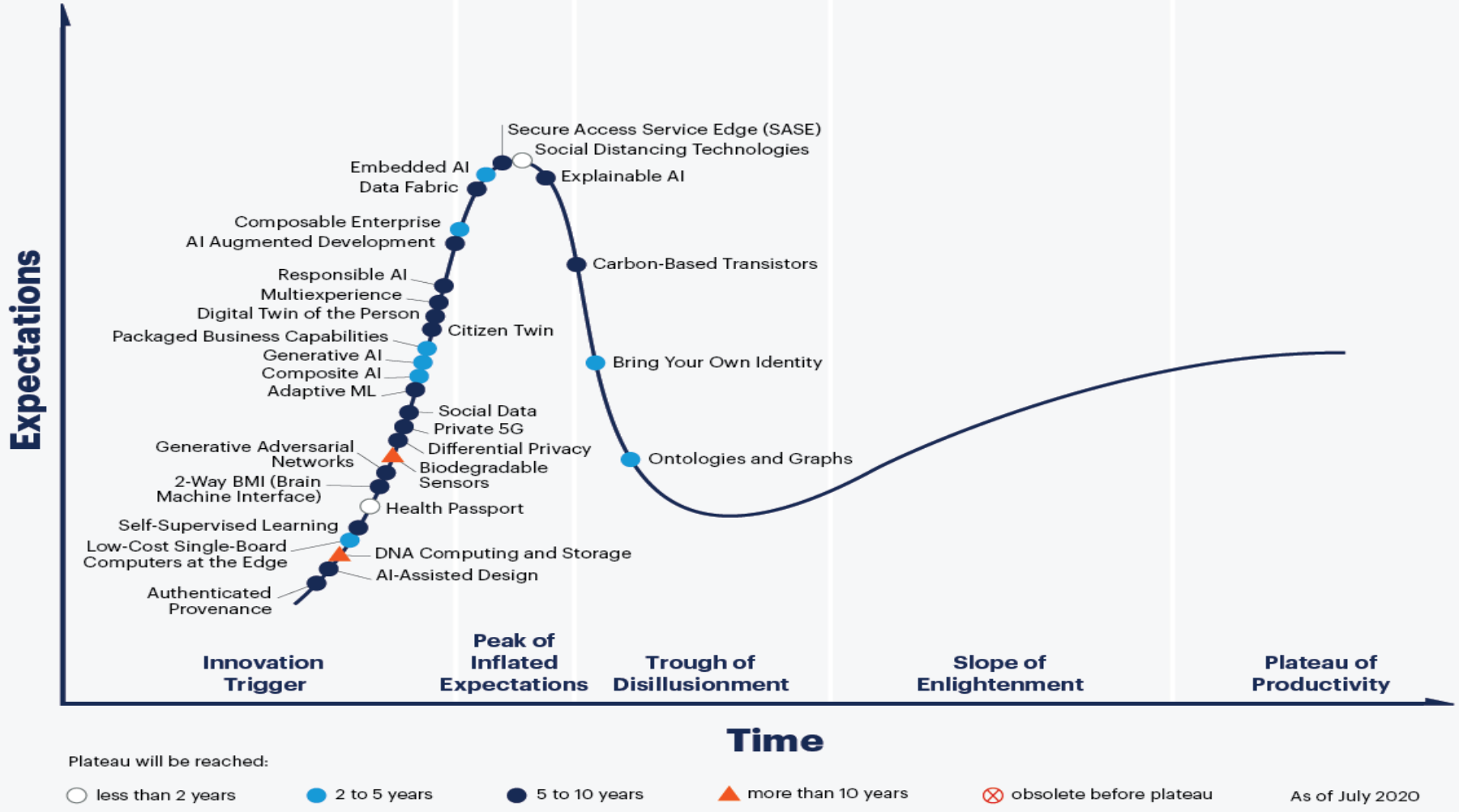
- Intro to unstructured data (NLP)
- Workshop (approx. 1-3)
- Case study 1

Emerging Technologies Hype Cycle 2012



Gartner

Hype Cycle for Emerging Technologies, 2020

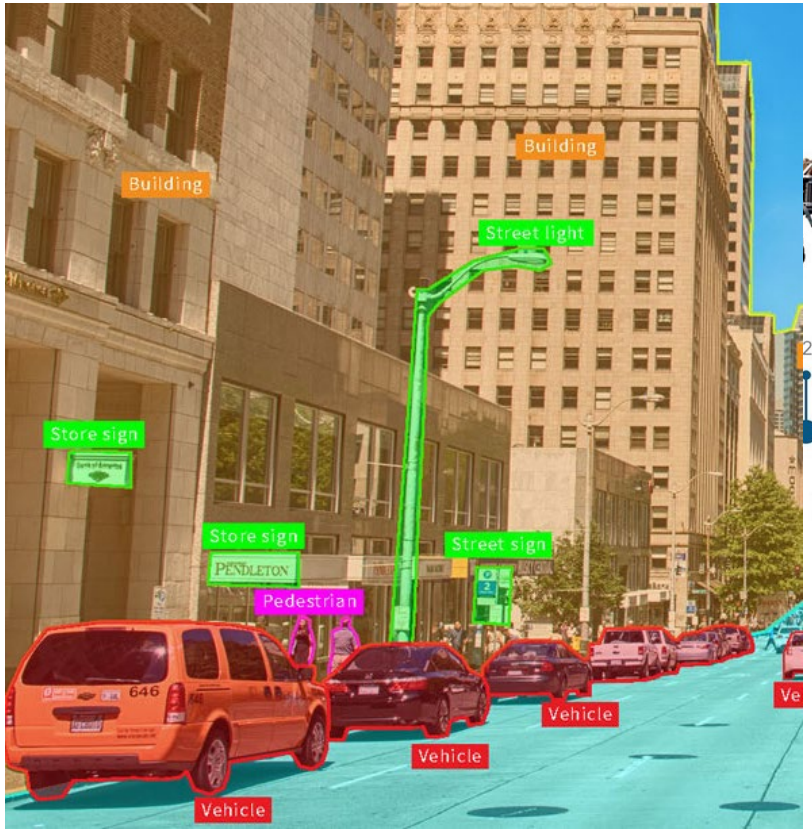


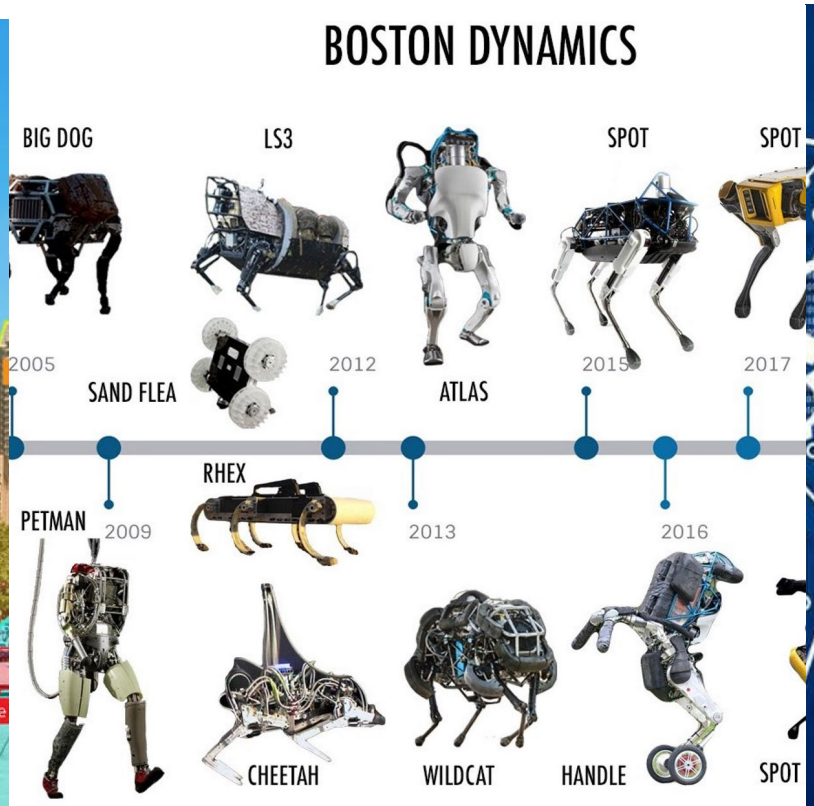
gartner.com/SmarterWithGartner

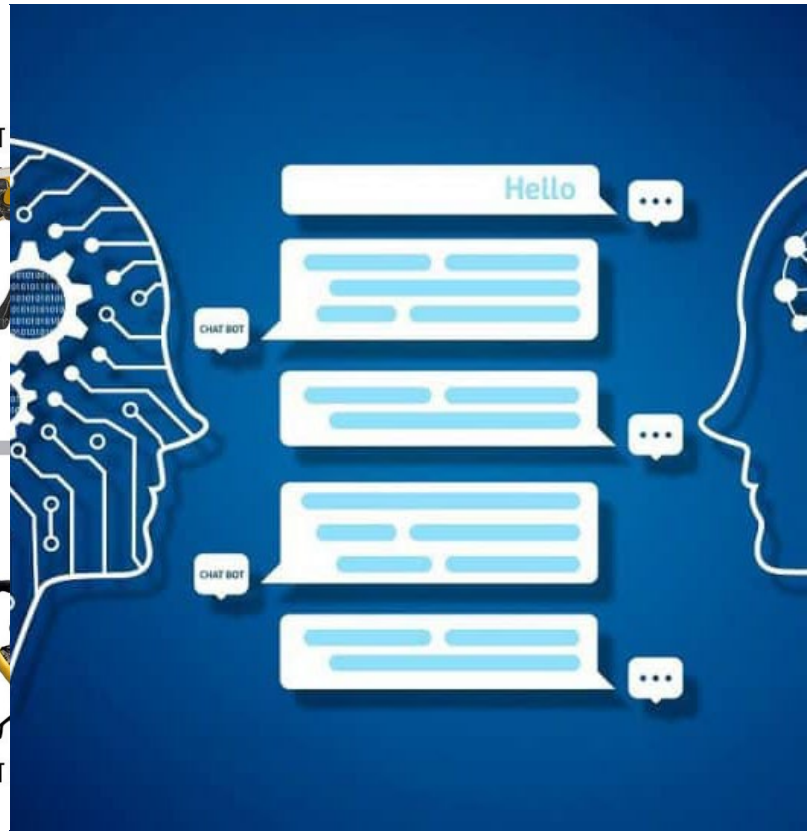
Source: Gartner

Gartner

Area of interest







Modern Artificial Intelligence (AI)

Introduction (cont.)

- Data continues to grow exponentially
  - Estimated to be 2.5 MTB a day
  - Grow to 40 BTB by 2020 (50 * of 2010)
- Approx. 80% of data is estimated to be unstructured/text-rich data
  - >4.5 billion web pages
  - >40 million articles (5 million in English)
  - >500 million tweets a day, 200 billion a year
  - >1.5 trillion queries on Google a year

Machine Versus Men

Machine Versus Men (cont.)

- Can machine beat the best of man in what man is supposed to be the best at?
- https://www.youtube.com/watch?v=YgYSv2KSyWg
- Watson, which is called DeepQA
- Watson had access to 200 million pages of structured and unstructured content consuming four terabytes of disk storage.

DeepQA overall architecture

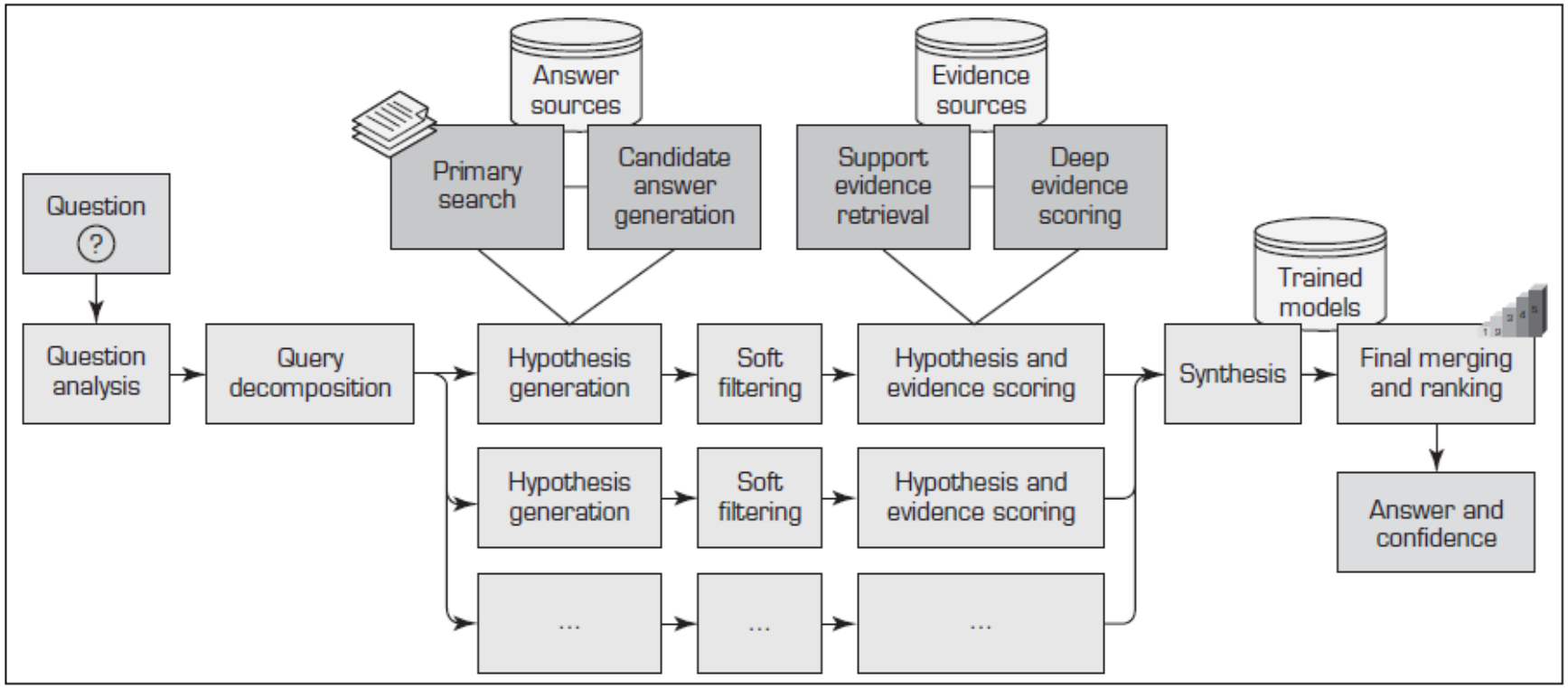


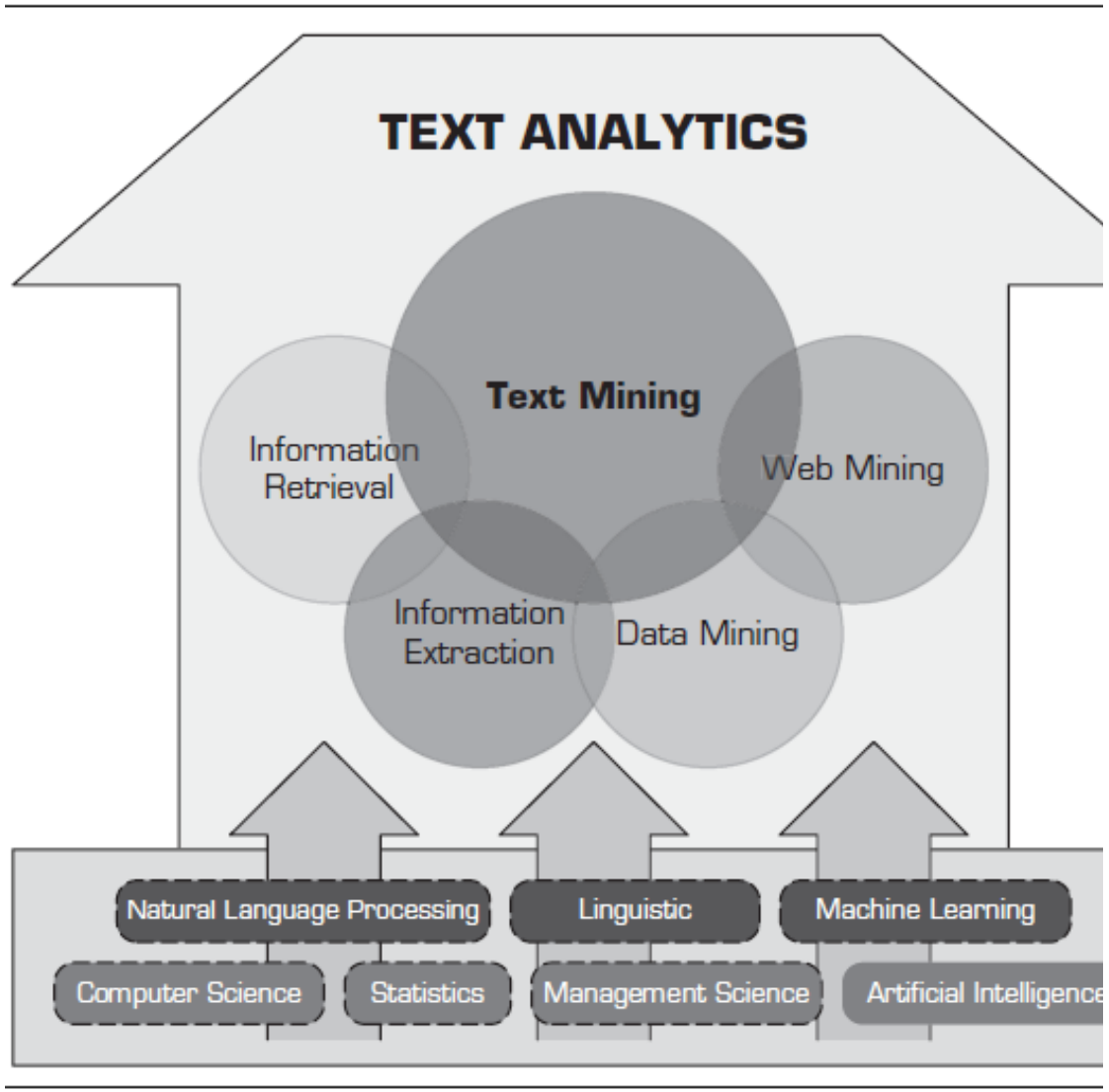
FIGURE 7.1 A High-Level Depiction of DeepQA Architecture.

Text Analytics

- The vast majority of business data is stored in text documents that are virtually unstructured.
- Text analytics is a broader concept that includes information retrieval (e.g., searching and identifying relevant documents for a given set of key terms) as well as information extraction, data mining, and Web mining,
- Whereas text mining is primarily focused on discovering new and useful knowledge from the textual data sources.

Text Analytics (cont.)

- Top-down approach vs bottom up
- Text Mining is a derivative of Data Mining.
- Sentimental Analysis is a derivative of Text Mining.



Text Mining

- AKA. Text data mining/knowledge discovery in textual database
- Large amount of unstructured data
- Word, PDF, XML, etc.

- Benefit domains:
  - In the areas where very large amount of textual data are being generated.
  - Could you give some example?

Text Mining (cont.)

- Law – court orders
- Academic research – research article
- Finance – quarterly report
- Medicine – discharge summaries
- Biology- molecular interactions
- Marketing – customer comments
- Social Science – Web board, Twitter , etc.
- Technology – e-mail? Is Gmail the smartest?

Text Mining applications

- **Information extraction** – identify the key phrases and relationship within text
- **Topic tracking** – predict/recommend other document of interest to user.
- **Summarization** – summarizing a document to save time of the reader.
- **Categorization** – identify the main themes of a document and put to the right themes
- **Clustering** – group similar documents without having a predefined set of categories
- **Concept linking** – connects related documents by identify their shared concepts.
- **Question answering** – find the best answer to a given question

Text Mining applications (cont.)

- **Intention mining/recognition/detection** – discover user intention based on comments, reviews, tweets, blogs
- **Concept mining** – extract idea and concept from large static social media
- **Sentiment Analysis** – categorize text to sentiment polarity (pos, neg, neu)

Text mining terminologies

- ***Unstructured data (versus structured data).*** *– human readable*
- ***Corpus*** *= dataset*
- ***Terms*** *– single word or multiword pharse from corpus*
- ***Concepts*** *– features generated from documents.*
- ***Stemming*** *– process of reducing inflected word to their root*
- ***Stop words-*** *words that are filtered out after processed.*
- ***Synonyms and polysemes*** – syntactically different/identical words
- ***Tokenizing*** *– block of text*
- ***Word frequency*** *- #time that word occur in the document*

Text mining terminologies (cont.)

- ***Morphology*** *– form and formation of words in a language*
- ***Morphemes*** *– an element in word*
- ***Word frequency*** *– the number of times a word is found*

Intro Natural Language Processing (NLP)

- Natural language vs Programming language
  - NL - human share information with human
  - PL – human tells machines what to do
- NLP – Machine can now process natural language (i.e., interpreter)

Intro NLP (cont.)

- Subfield of AI and Computation Intelligent/linguistics.
- Try to understanding the natural human language.
- Moving forward to syntax-drive (word counting) to true understanding and processing of NLP
- Considering grammatical, semantic constraint and context.

NLP practical applications

- **Editing** – spelling, grammar, style
- **Dialog** – Chatbot, assistant, scheduling
- **Email** – spam filter, classification, prioritization
- **Text mining** – Summarization, knowledge extraction
- **News** – event detection, fact checking, fake news detection
- **Attribution** – plagiarism detection, literacy forensics,
- **Creative writing** – Movie scripts, poetry, song lyrics.
- **Search**- web, documents, autocomplete
- **Chatbot** – Ambiguous commands, Q/A, scheduling

Intro Natural Language Processing (NLP)

- Bag-of-words (classical method)
  - Text, sentences, paragraph, or document -> words
  - Classification model <spam/legitimate>
  - One bag is filled with words found in spam messages (Viagra, stock, buy)
  - Another bag is filed with words related to user's friend or workplace.
- Human do not use words without some order or structure
  - Semantic and syntactic structure
- Text mining need to look for ways beyond the bag-of-words.

Challenge in NLP (non-practical)

- **Part of speech tagging (POS-tagging**)- identify Adverb verb, noun in the sentence.
- **Text segmentation** - Chinese/Thai/Other languages.
- **Word sense disambiguation** – a word may has more than one meaning.
- **Syntactic ambiguity** – grammar is ambiguous
- **Imperfect or irregular input** – typos , grammar errors

Workshop 1- Business understanding

- How would you apply NLP techniques or possible application into your interest business?

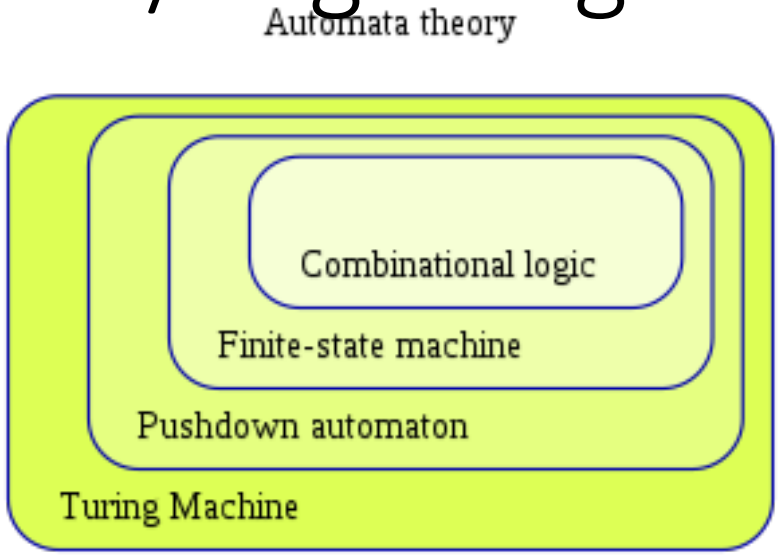
- Discuss, stakeholders, data sci talks, output, outcome, and impact.

Example

Activities	Outputs	Outcomes	Indicators
1. Retrieve data from Binance API and feature selection with the correlation method	Several variables of cryptocurrency price	Increasing of most relevant attributes for the predictions	The number of variables
2. Model selection and evaluation - Logistic Regression - Naive Bayes - K-Nearest Neighbors	The change of cryptocurrency price direction(moving up or down) as a prediction and several metrics evaluation	Increasing of prediction ability (accuracy), then choose the best model	Percentage of accuracy of each model
3. Hyperparameter tuning	Hyperparameter and its performance	Optimal Hyperparameter with best value of metric evaluation	Percentage of error of the model

Table 2: Problem Identification

Regular expression/Regular grammar



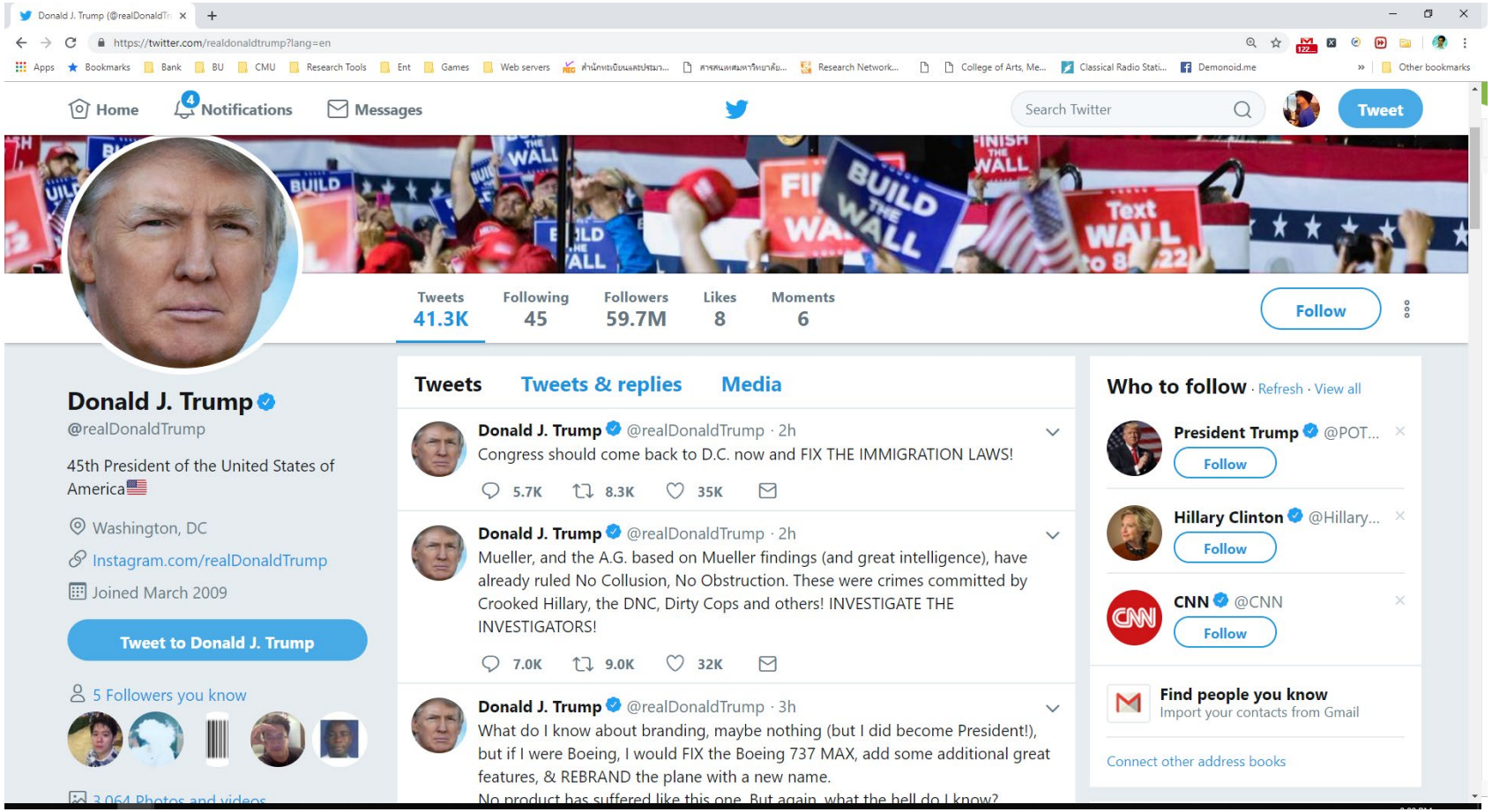
Finite State Machine (FSM) /Deterministic finite automation (DFA)

However, Natural Language ..

- Are not regular (rules)
- Are not context –free (pattern of string)
- Can’t be defined by any formal grammar.

Data hidden in text

https://github.com/preenet



Handling Text in Python

```
>>> txt1 = 'The Radical Left Democrats will never be satisfied with anything we give them. They will always Resist and Obstruct!’
>>> len(txt1)
116
>>> txt2 = txt1.split(' ');
>>> len(txt2);
19
>>> txt2
['The', 'Radical', 'Left', 'Democrats', 'will', 'never', 'be', 'satisfied', 'with', 'anything', 'we', 'give', 'them.', 'They', 'will', 'always', 'Resist', 'and', 'Obstruct!']
>>>
```

Handling Text in Python (cont.)

- **Finding long words: e.g. words that has more than 3 letters.**

```
>>> [w for w in txt2 if len(w) > 3]
['Radical', 'Left', 'Democrats', 'will', 'never', 'satisfied', 'with', 'anything', 'give', 'them.', 'They', 'will', 'always', 'Resist', 'Obstruct!’]
```

- **Finding capitalized words:**

```
>>> [w for w in txt2 if w.istitle()]
['The', 'Radical', 'Left', 'Democrats', 'They', 'Resist', 'Obstruct!']
```

Handling Text in Python (cont.)

Find words which ends with ‘s’:

```
>>> [w for w in txt2 if w.endswith('s')]
['Democrats', 'always’]
```

Find unique words: using set():

```
>>> txt3 = 'To be or not to be’
>>> txt4 = txt3.split(' ‘)
>>> len(txt4)
6
>>> len(set(txt4))
5
>>> set(txt4)
{'be', 'not', 'to', 'To', 'or'}
```

Handling Text in Python (cont.)

```
>>> set([w.lower() for w in txt4])
{'be', 'not', 'or', 'to'}
>>>
```

Handling Text in Python (cont.)

- **Word comparation in Python**
- s.startswith(t)
- s.endswith(t)
- t in s
- s.istitle()
- s.islower()
- s.isupper()
- s.isdigit(), s.isalnum(), s.isalpha()

String operations

- s.lower(); s.upper(); s.titlecase()
- s.splits()
- s.splitlines()
- s.join(t) //
- s.strip() // take out all the white space
- s.find(t)// find the specific substring
- s.replace(u, v)// u will be replace by v

Words to characters

```
>>> txt5 = 'ouagadougou'
>>> txt6 = txt5.split('ou')
>>> txt6
['', 'agad', 'g', ‘’]
>>> 'ou'.join(txt6)
'ouagadougou'
```

Words to characters (cont.)

```
>>> txt5.split('')
Traceback (most recent call last):
 File "<pyshell#7>", line 1, in <module>
    txt5.split('')
ValueError: empty separator
>>> list(txt5)
>> [c for in c txt5]
['o', 'u', 'a', 'g', 'a', 'd', 'o', 'u', 'g', 'o', 'u']
```

Cleaning text

```
>>> txt8 = '    A quick brown fox jumped over the lazy dog. '
>>> txt8.split(' ')
['', '', '\t', 'A', 'quick', 'brown', 'fox', 'jumped', 'over', 'the', 'lazy', 'dog.', '']
>>> txt8.strip()
'A quick brown fox jumped over the lazy dog.'
```

Changing text

```
>>> txt9 = txt8.strip()
>>> txt9
'A quick brown fox jumped over the lazy dog.'
>>> txt9.find('o')
10
>>> txt9.rfind('o')
40
>>> txt9.replace('o', 'O')
'A quick brOwn fOx jumped Over the lazy dOg.'
```

Handling larger texts

- **Reading files line by line**

```
>>> f = open('/Users/Pree/Desktop/notre dame.txt')
>>> f.readline()
'Notre-Dame de Paris, often referred to simply as Notre-Dame, is a medieval Catholic cathedral on the ÃŽle de la CitÃ© in the 4th arrondissement of Paris, France.\n'
```

- **Reading the full file**

```
>>> f.seek(0)
0
>>> txt12 = f.read()
>>> len(txt12)
```

Handling larger texts (cont.)

```
>>> txt13 = txt12.splitlines()
>>> len(txt13)


>>> txt13
['Notre-Dame de Paris, often referred to simply as Notre-Dame, is a medieval Catholic cathedral on the ÃŽle de la CitÃ© in the 4th arrondissement of Paris, France.', '', 'The cathedral is consecrated to the Virgin Mary and considered to be one of the finest examples of French Gothic architecture.']
>>> txt13[0]
'Notre-Dame de Paris, often referred to simply as Notre-Dame, is a medieval Catholic cathedral on the ÃŽle de la CitÃ© in the 4th arrondissement of Paris, France.'
```

File operations

- f = open(filename, mode)
- f.readline(); f.read(); f.read(n)
- for line in f: doSomething(line)
- f.seek(n)
- f.write(message)
- f.close() // close the file handler
- f.closed// check if the file close

Processing free-text

>>> txt10 = '#DrainTheSwamp - @GreggJarrett: No one takes anything Schiff says seriously because he lost all credibility. For 2 years he claimed there was a mound of criminal evidence. Where is it? Show us… because it doesn’t exist. #MAGA #AmericaFirst #Dobbs’

How to find out the callouts and hashtags?

Finding specific words

Hashtags

```
>>> [w for w in txt10.split() if w.startswith('#')]
['#DrainTheSwamp', '#MAGA', '#AmericaFirst', '#Dobbs’]
```

Callouts

```
>>> w for w in txt10.split() if w.startswith('@')]
['@GreggJarrett:', '@']
```

Finding patterns with regular expressions

- Callouts are more than just tokens starting with ‘@’
- @username @UK_Spokesperon
- Match something after ‘@’
  - Alphabets
  - Numbers
  - Special symbols such as ‘_’

Finding patterns with regular expressions

Callouts

```
>>> w for w in txt10.split() if w.startswith('@')]
['@GreggJarrett:', '@']
```

Import regular expressions first

```
import re
[w for w in txt10.split(' ') if re.search('@[A-Za-z0-9_]+', w)]
['@GreggJarrett:']
```

Parsing the callout regular expression

- @[A-Za-z0-9_]+
- Starts with @
- Followed by any alphabet (upper or lower case), digit, underscore
- That repeats at least once, but any number of times

Metacharacters

Metacharacters are characters with a special meaning:

Character	Description	Example
[]	A set of characters	“[a-m]”
\	Signals a special sequence	“\d”
.	Any character(except newline character)	“he..o”
^	Starts with	“^hello”
$	Ends with	“world$”
* (repetitions)	Zero or more occurrences	“aix*”

Metacharacters

- \d – any digit
- \D – any non-digit
- \s – any white space char, [\t\n\r\f\v]
- \S – opposite the above
- \w – Alphanumeric character , [a-zA-Z0-9_]
- \W – [^ a-zA-Z0-9_]

Sets

A set is a set of characters inside a pair of square brackets

Set	Description
[arn]	Returns a match where one of the specified characters (a, r, or n) are present
[a-n]	Returns a match for any lower-case character, alphabetically between a and n
[^arn]	Returns a match for any character EXCEPT a, r, and n
[0123]	Returns a match where any of specified digits(0, 1, 2, or 3) are present
[0-9]	Returns a match for any digit between 0 and 9
[0-5][0-9]	Returns a match for any two-digit number from 00 and 59
[a-zA-Z]	Returns a match for any character alphabetically between a and z, lower case OR upper case

Example 1 – search() function

The search() function searches the string for a match, and returns a Match object if there is a match.

If there is more than one match, only the first occurrence of the match will be returned:

```
import re

#Check if the string starts with "The" and ends with " ChiangMai":

txt = "The rain in ChaingMai"
x = re.search("^The.*ChiangMai$", txt)

if (x):
  print("YES! We have a match!")
else:
  print("No match")
```

Example 2 – findall() function

- The findall() function returns a list containing all matches.
- import re

#Return a list containing every occurrence of "ai":

```
str = "The rain in Chaing Mai"
x = re.findall("ai", str)
print(x)
['ai', 'ai']
```

Example 2.1 findall() function

- Finding specific characters

```
>>> txt12 = 'ouagadougou'
>>> re.findall(r'[aeiou]', txt12)
['o', 'u', 'a', 'a', 'o', 'u', 'o', 'u’]
>>> re.findall(r'[^aeiou]', txt1)
['g', 'd', 'g']
```

Example 3 – sub() function

- The sub() function replaces the matches with the text of your choice:

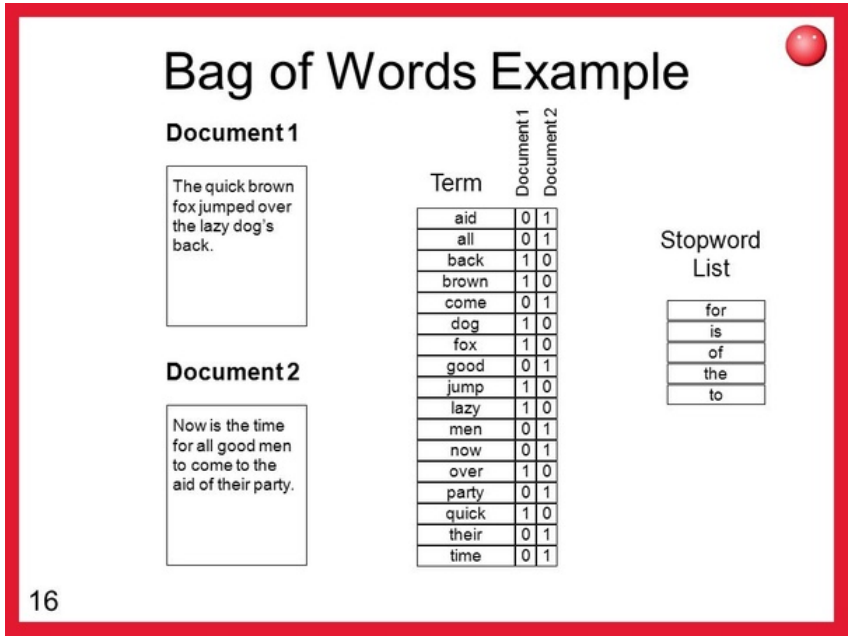
```
import re

str = "The rain in Chaing Mai"
x = re.sub("\s", "9", str)
print(x)
```

The9rain9in9Chiang Mai

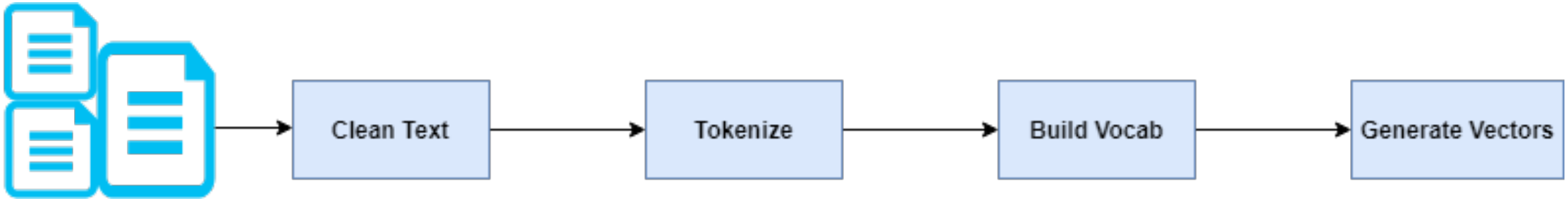
Bag of Words (BOW)

- Machine learning can’t work with raw text
- Text must be convert to numbers, vectors of numbers
- Usually this is called “feature extraction”
- Bag of words is the most simple approach.
- Represent of string based on frequency of entities



Text pre-processing pipeline with BOW

- Clean the text (white space, upper case, etc.)
- Tokenizing
- Build dictionary
- Filter is also useful (sorting histogram and take top 50)
- The vectors will be use in ML algorithms for document classification or clustering.



BOW modeling in action

```
wordfreq = {} # create dict and iterate through each sentence
for sentence in corpus:
    tokens = nltk.word_tokenize(sentence)
    for token in tokens:
        if token not in wordfreq.keys():
            wordfreq[token] = 1
        else:
            wordfreq[token] += 1

# filter dimension to 300
import heapq
most_freq = heapq.nlargest(300, wordfreq, key=wordfreq.get)

sentence_vectors = [] # create sentence list and iterate throgh each sentence
for sentence in corpus:
    sentence_tokens = nltk.word_tokenize(sentence)
    sent_vec = []
    for token in most_freq:
        if token in sentence_tokens:
            sent_vec.append(1)
        else:
            sent_vec.append(0)
```

```
1 from sklearn.feature_extraction.text import CountVectorizer
2 matrix = CountVectorizer(max_features=300)
3 X = matrix.fit_transform(data).toarray()
```

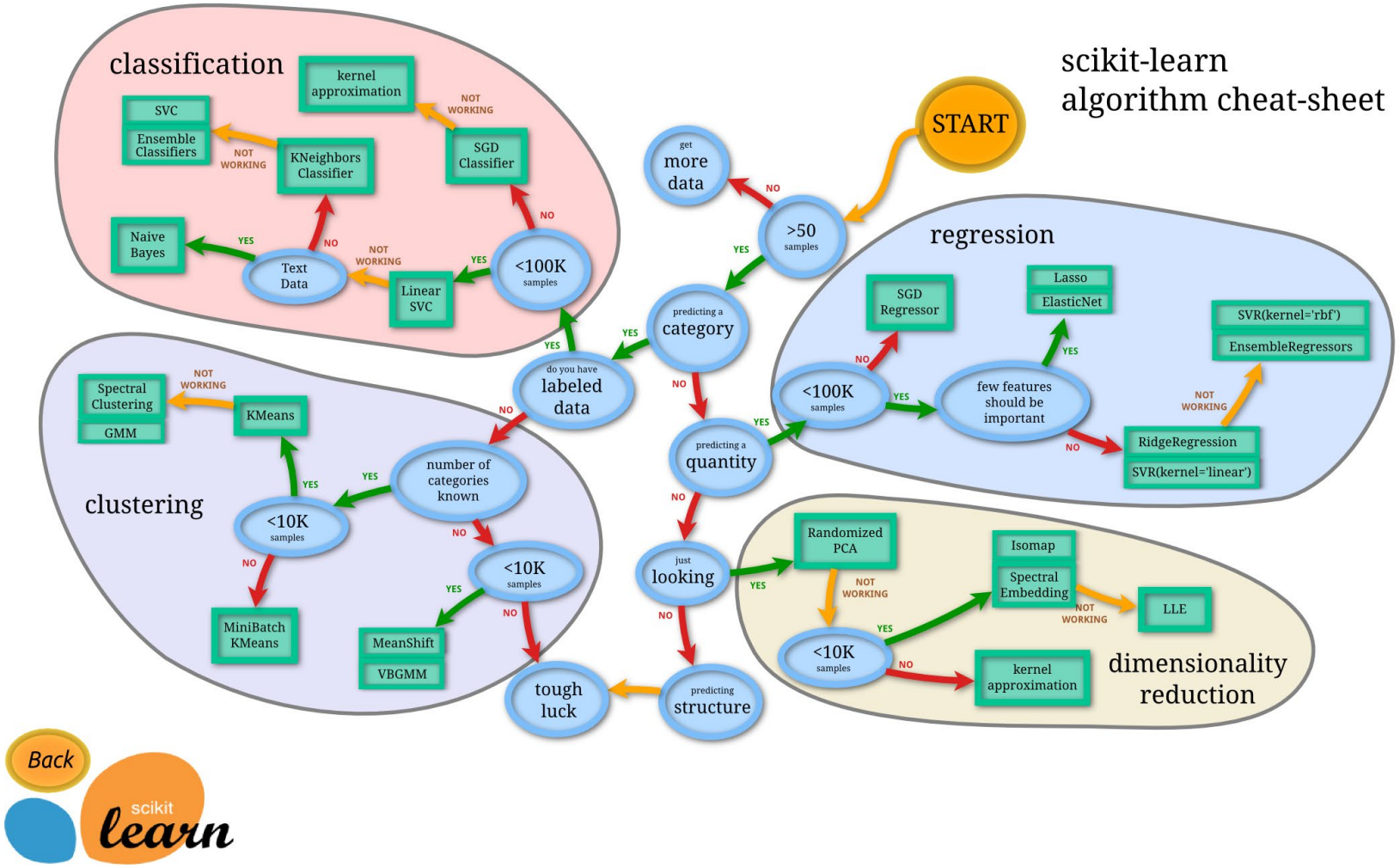
Workshop 2 – Basic text preprocessing

What we have learned so far

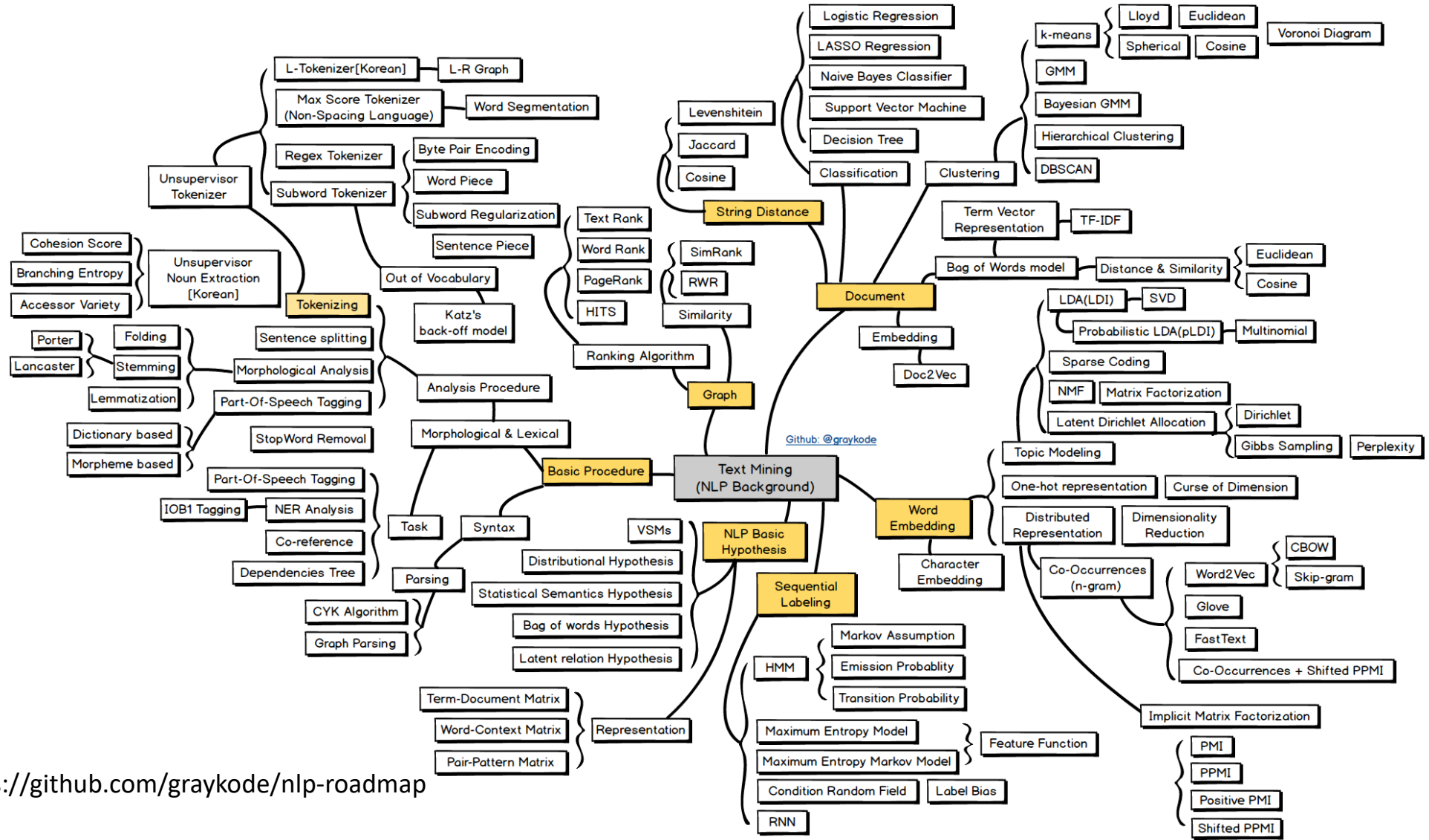
- Intro to Text Analytic
- Data pre-processing
  - Handling text sentences
  - Splitting sentences into words
  - Splitting words into characters
  - Finding unique words
  - Intro to Regular Expression and operations
  - Words to vectors
  - Word tokenization
  - Feature extraction
- BOW Modelling

- ฉันจะไปโรงเรียนนะจ๊ะ

Recall: There are many algorithm to try

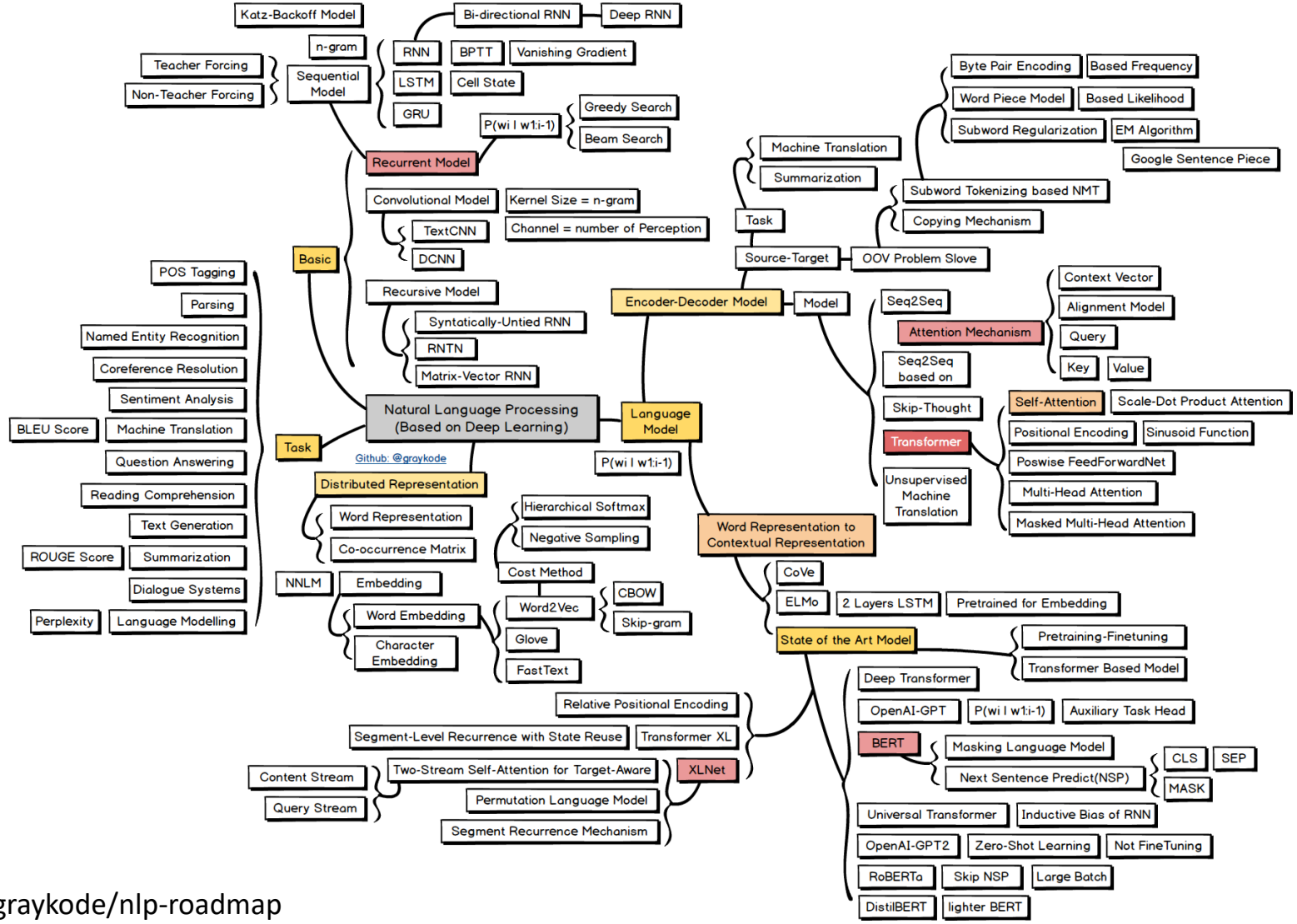


NLP road map



Source:https://github.com/graykode/nlp-roadmap

NLP road map (cont.)



Source: https://github.com/graykode/nlp-roadmap

Case study 2021 / NLP application in Action

- AI chat bot with Stress detection
- Read more at:
  - https://arxiv.org/abs/1911.00133

Conversational Agents

- AKA. Dialogue System, Dialogue Agents, Chatbots
- Personal Assistants on phones or other platforms
  - Alexa, SIRI, Google Assistant, Cortana
- Playing music, setting times and clock
- Chatting for fun
- Booking, scheduling reservation
- Clinical uses for mental heath (like my project)

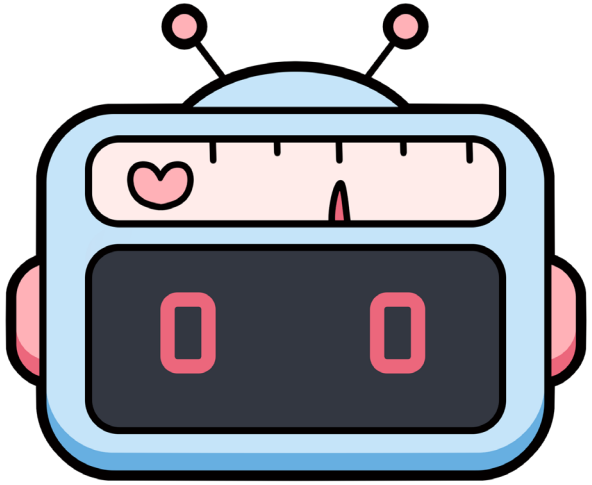
Conversational Agents (cont.)

- Chatbots
  - Mimic informal human chatting
  - For fun, even for therapy
- Task-based
  - Personal assistant
  - Book seat in restaurant, movie theater, flights

Chatbot Arch.

- Rule-based
  - Pattern-action rules (ELIZA)
- Corpus-based(data-driven)
  - Information Retrieval
  - Model-based (My chatbot)

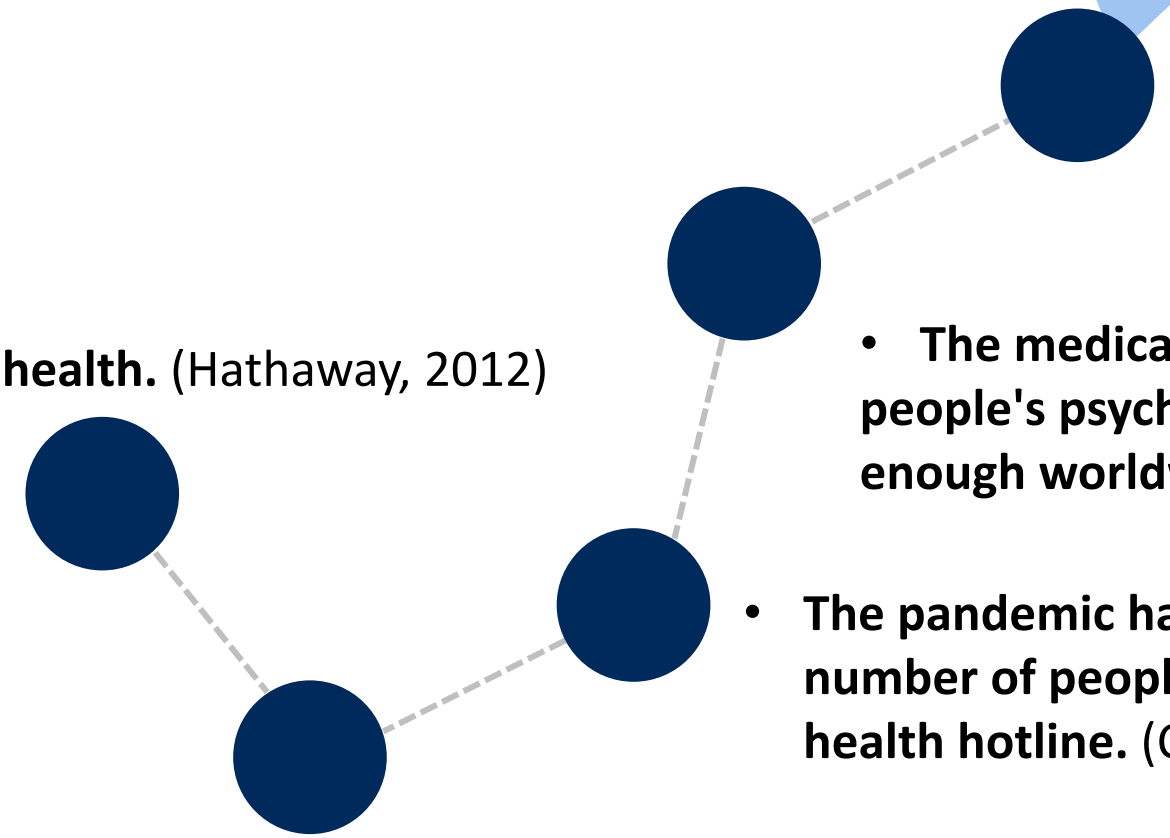
AI Chatbot with Stress Detection



Group name: *FTW_SD*
Presenter: *Pakin Kampeera (612115005)*
Aoxue Gui (612115501)
Project Advisor: *Dr. Pree Thiengburanathum*

Background

- **Factors that affect people's active medical treatment:**
  - **Cost**
  - **Inconvenience of making an appointment**
  - **Obstacles to confiding in strangers**



- **Stress hurts human health.** (Hathaway, 2012)
- **The medical expenses invested to alleviate people's psychological problems are far from enough worldwide.** (United Nations, 2019)
- **The pandemic has caused a surge in the number of people calling the mental health hotline.** (CDC, 2020)
- **Most people have stress problems.** (Mental Health Foundation, 2018; APA, 2020; United Nations, 2019; KFF, 2020)

Motivation

we hope to develop a "self-help" tool to let limited psychological counselling services help those who need it more

Aim

Develop a chatbot to detect the user stress during communication and a dashboard to visually shows the statistical data.

Feature Overview

05.

Data cleaning, Data preprocessing, Data analysis

- Analyze sentences.

06.

Report generate

- View own stress at last 6 months information in line chart.

07.

Notification

- View notification message.

Feature Overview

05.

Data cleaning, Data preprocessing, Data analysis

- Analyze sentences.

06.

Report generate

- Generate intuitive report to see stress change during the last six month.

07.

Notification

- Send notification message.

Feature Overview

01.

Authentication

- Register
- Login
- Logout
- Reset password

02.

Dashboard

- View statistic data
- View sentences table
- Export table to CSV

03.

User management

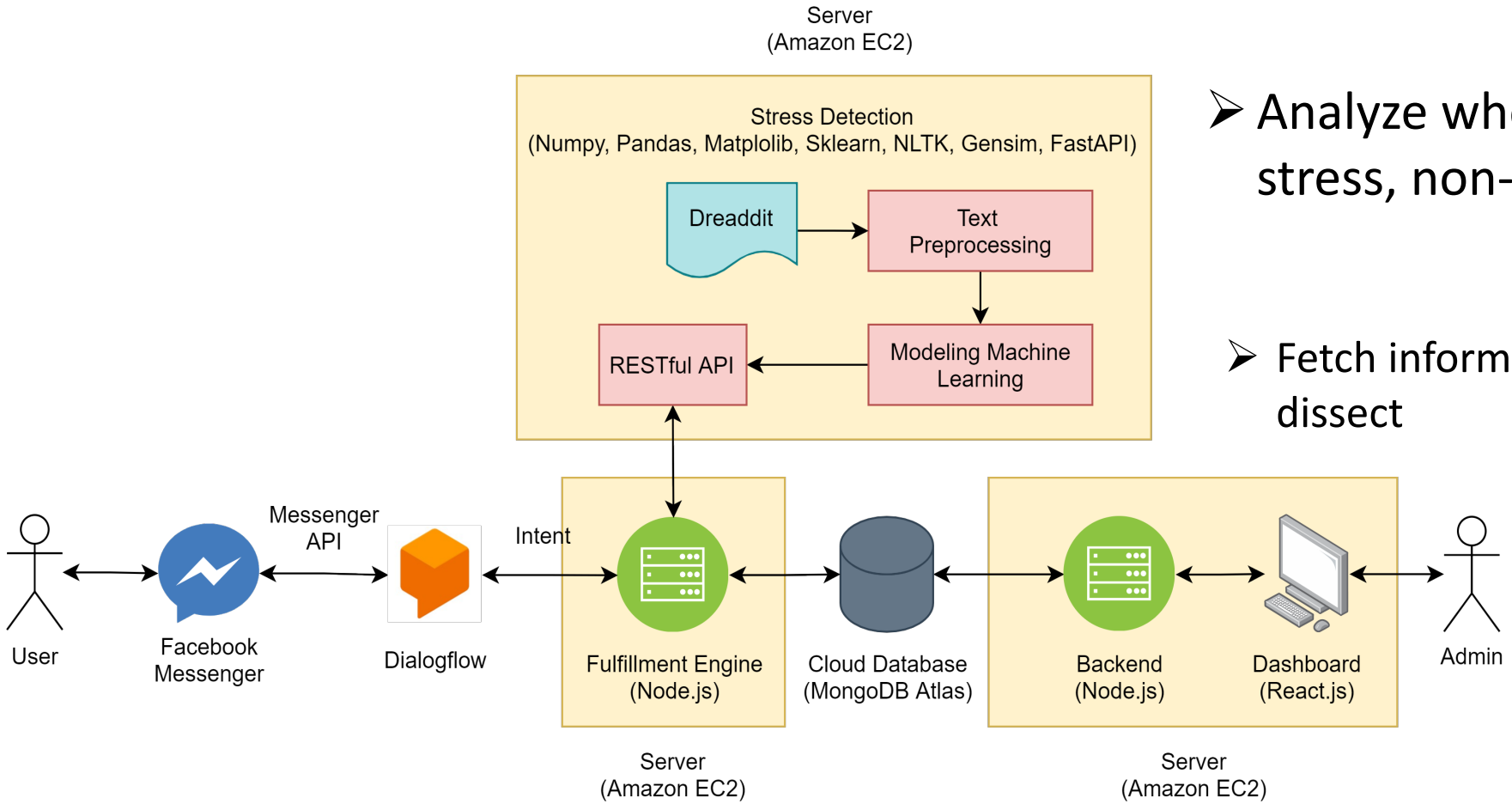
- View account information
- Change user

04.

Facebook messenger Chatbot

- Search
- Communic username or ate with message Chatbot

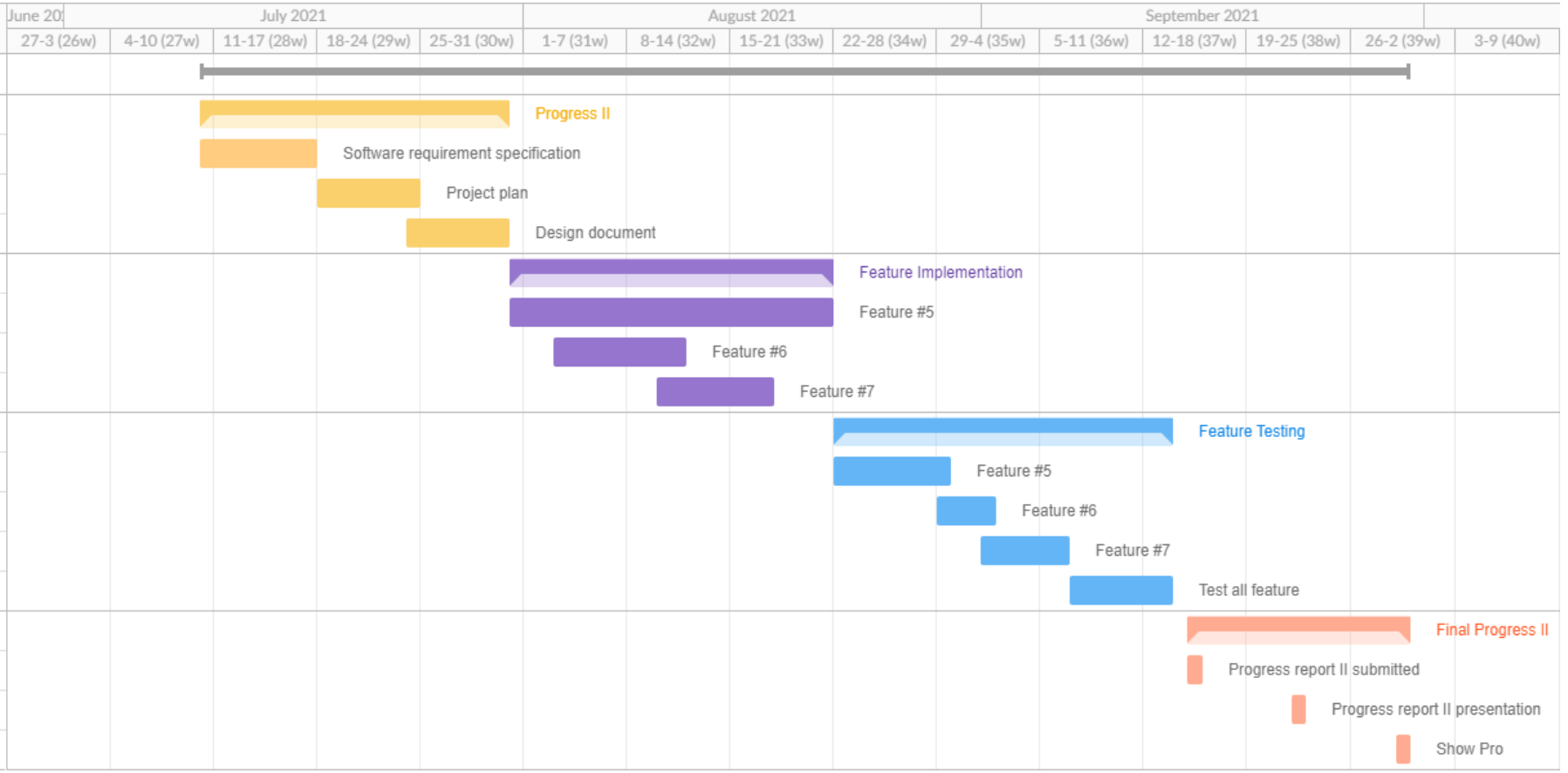
Product Perspective



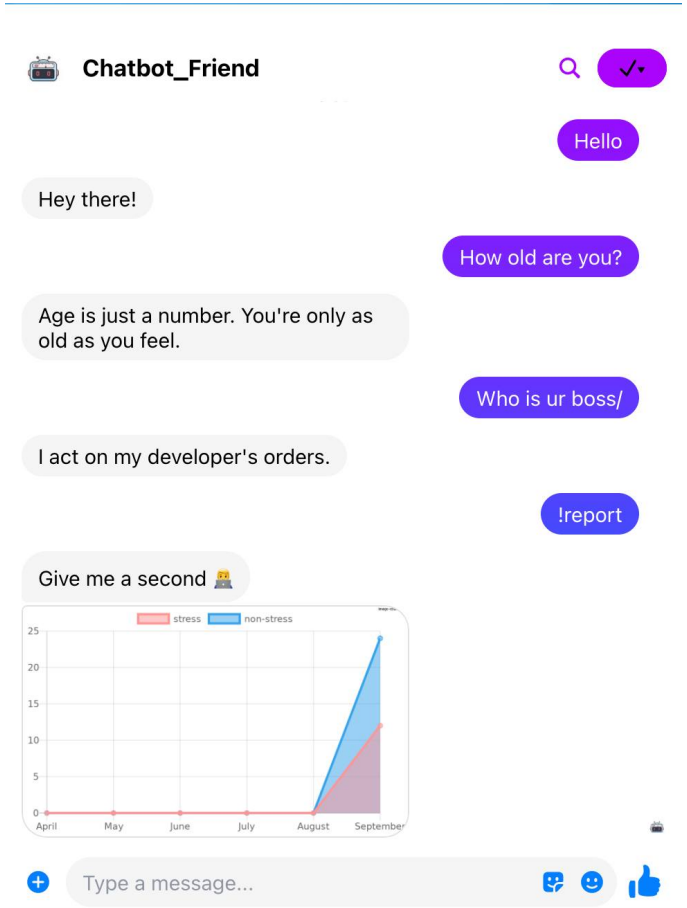
- Analyze whether the sentences are stress, non-stress and cannot tell
- Fetch information from database to dissect

Milestone

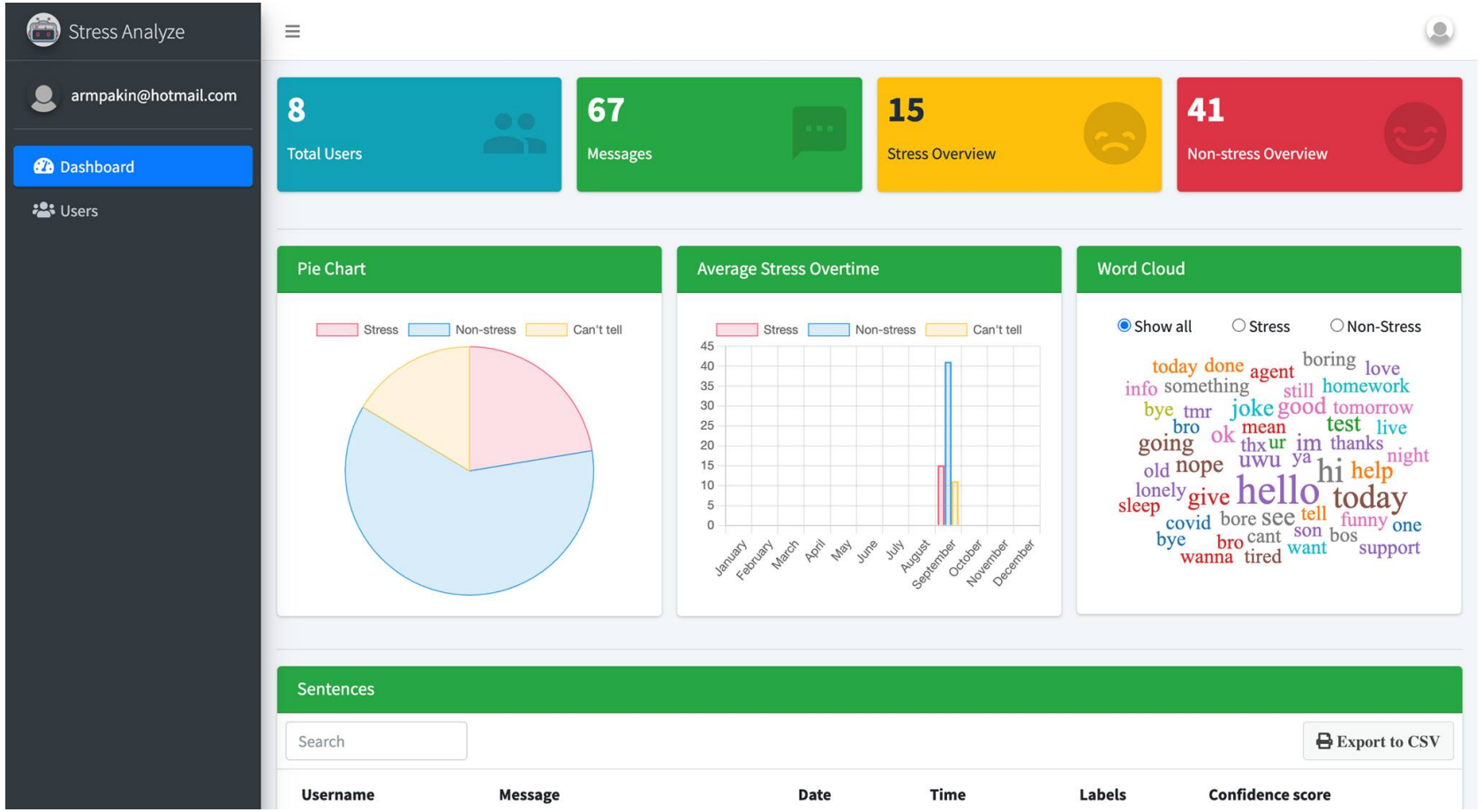
	Task name	Start date	End date
		10/07/2021	**29/09/2021**
1	**Progress II**	**10/07/2021**	**30/07/2021**
1.1	Software requirement specification	10/07/2021	17/07/2021
1.2	Project plan	18/07/2021	24/07/2021
1.3	Design document	24/07/2021	30/07/2021
2	**Feature Implementation**	**31/07/2021**	**21/08/2021**
2.1	Feature #5	31/07/2021	21/08/2021
2.2	Feature #6	03/08/2021	11/08/2021
2.3	Feature #7	10/08/2021	17/08/2021
3	**Feature Testing**	**22/08/2021**	**13/09/2021**
3.1	Feature #5	22/08/2021	29/08/2021
3.2	Feature #6	29/08/2021	01/09/2021
3.3	Feature #7	01/09/2021	06/09/2021
3.4	Test all feature	07/09/2021	13/09/2021
4	**Final Progress II**	**15/09/2021**	**29/09/2021**
4.1	Progress report II submitted	15/09/2021	15/09/2021
4.2	Progress report II presentation	22/09/2021	22/09/2021
4.3	Show Pro	29/09/2021	29/09/2021



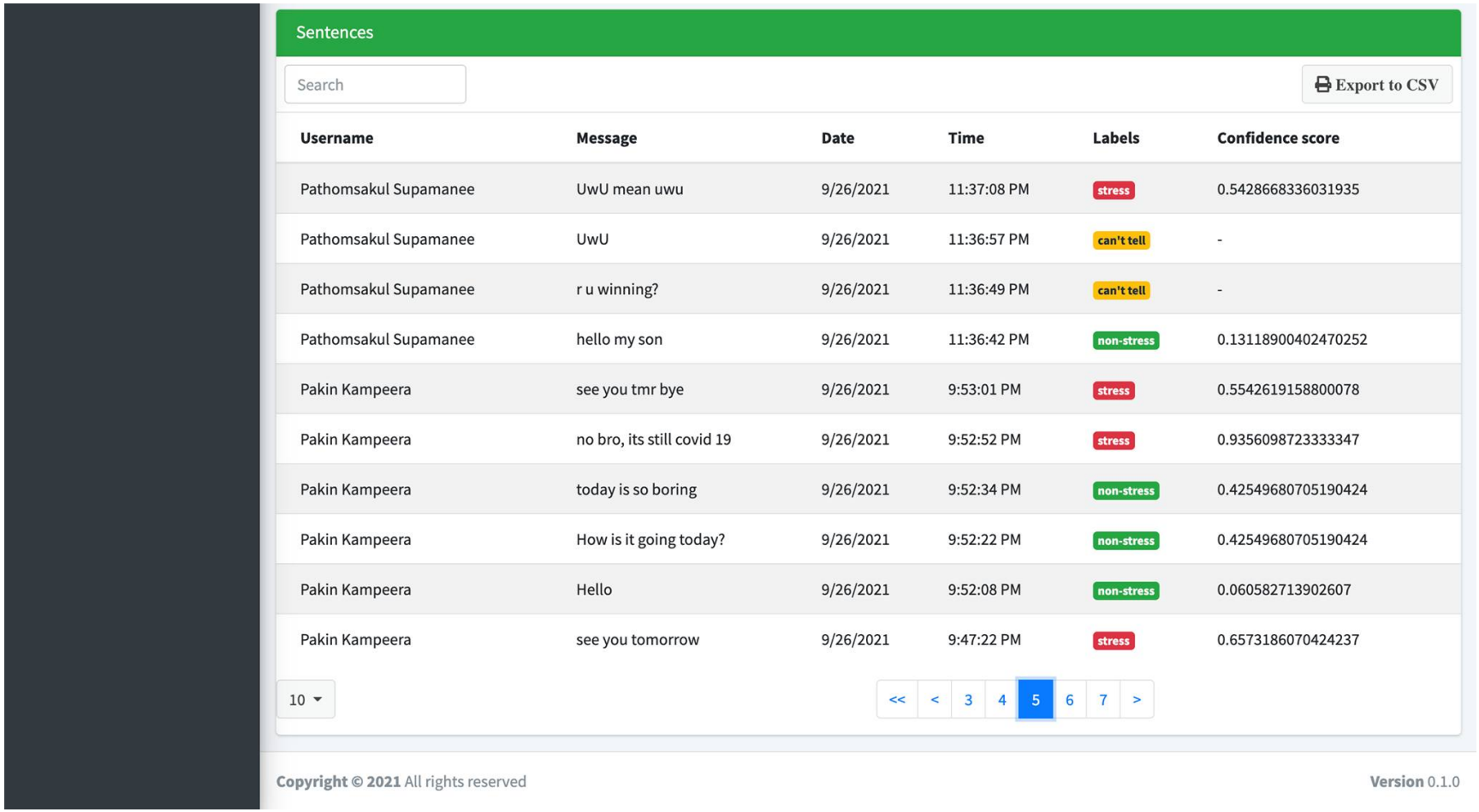
User interface



User interface



User interface



Lead out: more projects.

References

- Lane, H., Hapke, H., & Howard, C. (2019). *Natural Language Processing in Action: Understanding, analyzing, and generating text with Python* (1st edition). Manning.
- Turban, E., Delen, D., & Sharda, R. (n.d.). *Business Intelligence, Analytics, and Data Science: A Managerial Perspective*.
- Turcan, E., & McKeown, K. (2019). Dreaddit: A Reddit Dataset for Stress Analysis in Social Media. *ArXiv:1911.00133 [Cs]*. http://arxiv.org/abs/1911.00133