Twitter Semantic Analysis and Application on CoronaVirus COVID-19 Dataset

Katarzyna Pasternak

Abstract

In the last couple years, the use of personal virtual assistants such as Apple's Siri or Amazon's Alexa became very popular and the demand of its usage spread across disciplines. Also the concept of personal robot assistant at home has been introduced and currently is researched. In this project, I propose creating a topic specific voice assistant specializing on the topic of Novel CoronaVirus COVID-19 and implementing it on the Toyota Human Support Robot (HSR). The information for the assistant is to be obtained from Twitter posts, analyzed and applied as the model and base knowledge for the assistant.

1. Introduction

Data mining allows for extracting patters form large amount of data in a dataset or many datasets combined. In the process different methods are used within two large categories - prediction and description. Under predictive tasks, classification, prediction and time-series analysis can be distinguished, while in descriptive tasks we specify association, clustering and summarization. For the purpose of this project, I will perform classification and association on car_evaluation dataset [2].

RoboCup is an annual international robotics competition founded in 1996 with the aim of promoting robotics and AI research, by offering publicly appealing but formidable challenges [6]. The teams in the RoboCup@Home League are challenged with programming service robots such as the Toyota Human Support Robot (HSR) [11] (ref. Figure 1) to assist humans with household tasks.

Classification derives a model to determine the class of an object based on its attributes [5]. A collection of records will be available, each record with a set of attributes. One of the attributes will be class attribute and the goal of classification task is assigning a class attribute to new set of records as accurately as possible.

Figure 1. The Toyota HSR Robot.

2. Instructions

Please follow this simple instructions to compile the code in the .zip folder. Run the code using `'python3'` python file name depending on your python installation (you want to be using python3.

To run the speech assistant on your machine please refer to http://wiki.ros.org/ROS/Tutorials. I provide the whole catkin workspace but feel free to just copy the slang package to your workspace. To run the program source the workspace by `source ~/catkin_ws/devel/setup.bash`. On the separate terminal run `roscore`. Make sure you are in the `~catkin_we\src\` directory. Ensure that `test_weather.py` is executable. Then run `rosrun tweet_semantic test_weather.py`. This requires a lot of dependencies.

2.1. Tools Used and Requirements

The code is written in Python [9] using Pyton 3.6 to compile. The code depends on the following libraries: SciPy [10], NumPy[3], Sklearn [1, 7], Mlxtend [8], and Matplotlib [4].

The following were used for the speech application portion in this project: Robot Operating System (ROS), Kinetic version, Rasa NLU, Google CloudSpeech Chatito, Python 2 Speech Recognition API. ROS is a set of middleware libraries for robots. It enables users to develop reusable packages and services to handle specific tasks in robotics such as speech recognition, navigation, and mapping. ROS is widely used in both academia and industy, and can operate on various robotics platforms. For this project, we developed a package that allows users to ask a robot questions about the location of an item in a house.

To convert user questions from speech to text and the program's responses from text to speech, I used services provided by Google Cloud. Chatito was used to generate training examples of questions that the program might expect to hear from users, and the Python 2 Speech Recognition API was used to enable the program to "listen" for the user's questions. The reason why I used Python 2 instead of Python 3 here, even though Python 2 is no longer being updated as of 2020, is because I are using an older version of ROS called ROS Kinetic, which is written using a mix of C++ and Python 2.

3. Identification of Attributes

Both datasets comes with the following attributes : Tweet Id, Text, Name, Screen Name, UTC, Created At, Favorites, Retweets, Language, Client, Tweet Type, URLs, Hashtags, Mentions, Media Type, Media URLs. The attributes that will be used are Text and Language. I will also create an additional attribute that will become the classifier for each dataset.

For the opinion dataset, I use a sentiment classifier (`'sentiment_blob'`) to decide if the sentiment is positive, negative or neutral. For the professional dataset (CDC posts), I create a tag classifier (`'topic'`) to determine the topic of the post to be one of tips, vaccine, report, statistics or other.

4. Procedures

For the classification learning task, I used the random tree classifier provided in the SKLean library. Following are the procedures I used to prepare data and use it for training and testing.

4.1. Data obtaining

First, the datasets were imported from Twitter using TwitterAPI for Developers and converted into `.csv` format. The datasets `opinion.csv` and `CDCgov_user_tweets.csv` file using read_csv() function from Pandas. The dataset `opinion.csv` has 2080 entris and `CDCgov_user_tweets.csv` has 3177.

4.2. Data Pre-processing

The following procedures were performed to clean the datasets:

- formatting text into single line,
- removing text not in English (for opinion dataset),
- removing unrelated posts (CDC dataset only),
- remove links, punctuation, all single char, and prefixed 'b' if bite data,
- merging multiple spaces to single space,
- remove stop words, common words, rare words,
- spell correction and lower-casing,
- emoji removal (for CDC dataset),
- verbalize emoji emoticons (for opinions only).

4.3. Training and Testing

Before training on the dataset, the text string had to be vectored and given numerical representation using bag of words. For testing purposes we split each dataset into 70 training/30 testing subsets. As a text classifier, I used 'RandomForestClasifier' on the training set with 200 estimators and random state equal 0. I also computed the semantic/topic predictions and predictions confidence to use for data analysis.

5. Data Analysis

5.1. Opinion Dataset

The training dataset takes 70% of the entries and testing uses the remaining 30%. In the below figure [Figure 2], we can see the percentage distribution of sentiments as well as the calculated confidence of each sentiment being the correct representation of the text. In figure [Figure 3] the ac-

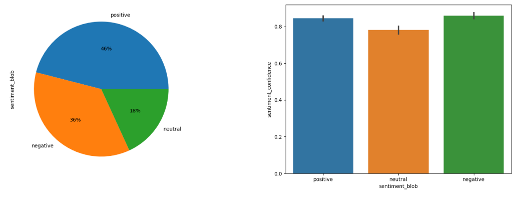
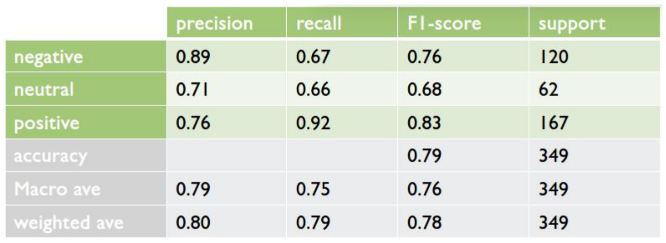


Figure 2. Sentiment distribution and confidence.

curacy score is 79% percent which is good enough for the purpose of the project but definitely is worth considering improving upon using other classification methods. In figure [Figure 4] we get the confusion matrix that shows pretty accurate prediction similarly as in Figure 2 graph.



	precision	recall	F1-score	support
negative	0.89	0.67	0.76	120
neutral	0.71	0.66	0.68	62
positive	0.76	0.92	0.83	167
accuracy			0.79	349
Macro ave	0.79	0.75	0.76	349
weighted ave	0.80	0.79	0.78	349

Figure 3. Result table.

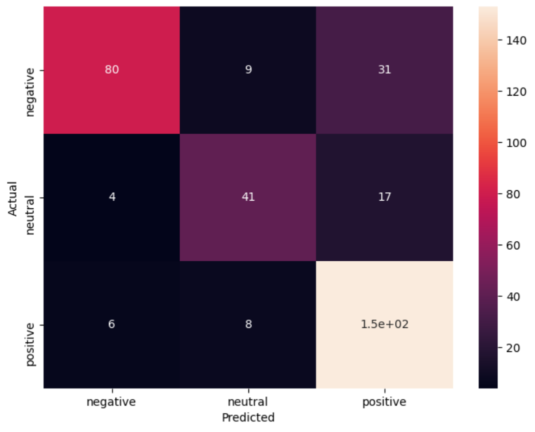


Figure 4. Confusion matrix.

5.2. CDC (Professional) Dataset

The training dataset takes 70% of the entries and testing uses the remaining 30%. In the below figure [Figure 6], we can see the percentage distribution of sentiments leans heavily towards topic other while other topics are spread quite evenly across tweets. In the figure [Figure 7] we see the accuracy score is quite high with 98%. This is probably the result of simplified if-then rule used and in order to obtain better results should be replaced with better topic tagging approach. In figure [Figure 8] we get the confusion matrix that shows pretty accurate prediction similarly as in Figure 6 graph.

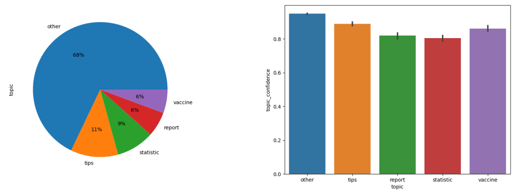
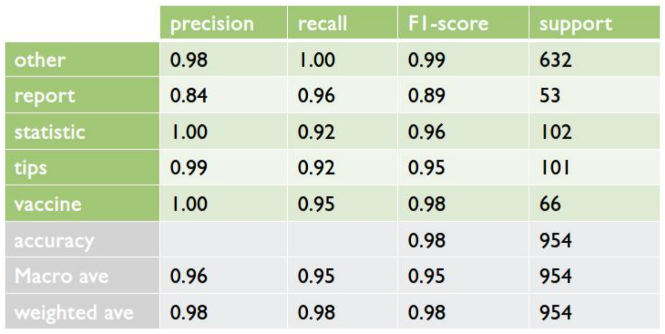


Figure 5. Topic distribution and confidence.



	precision	recall	F1-score	support
other	0.98	1.00	0.99	632
report	0.84	0.96	0.89	53
statistic	1.00	0.92	0.96	102
tips	0.99	0.92	0.95	101
vaccine	1.00	0.95	0.98	66
accuracy			0.98	954
Macro ave	0.96	0.95	0.95	954
weighted ave	0.98	0.98	0.98	954

Figure 6. Result table.

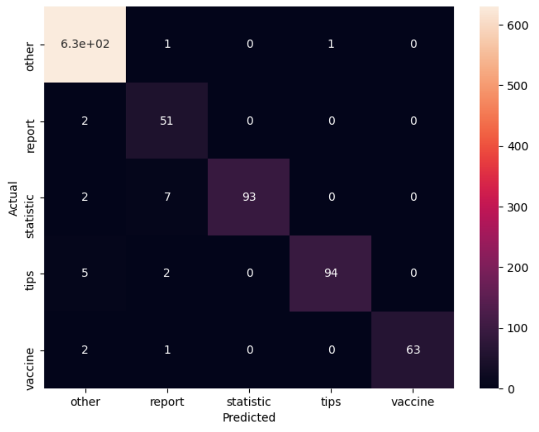


Figure 7. Confusion matrix.

6. Practical Application

A speech personal assistant was created to show the practical application of the processed data. The following shortly describes the process of using the data.

6.1. Question Dataset Creation and Training

The training and testing datasets were generated using Chatito according to the model in the figure. Only training dataset with (8000 entries) is necessary for the purpose

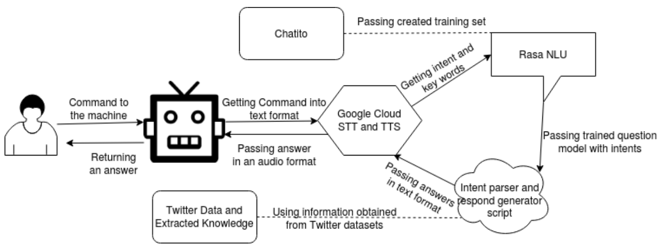
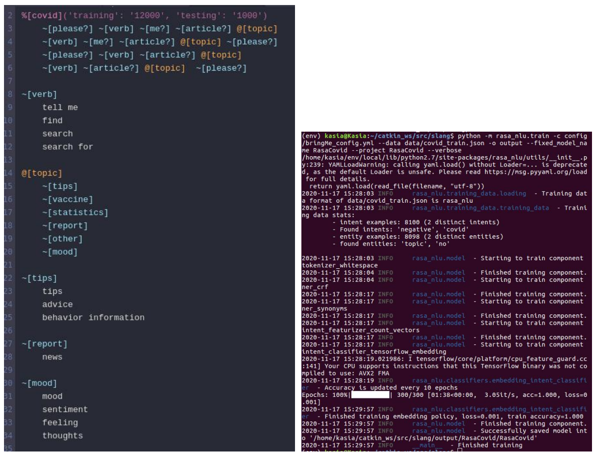

Chatito
Passing created training set
Rasa NLU
Getting intent and key words
Command to the machine
Getting Command into text format
Google Cloud STT and TTS
Passing trained question model with intents
Returning an answer
Passing answer in an audio format
Passing answers in text format
Intent parser and respond generator script
Twitter Data and Extracted Knowledge
Using information obtained from Twitter datasets

Figure 8. Application workflow.

of this program. Rasa uses Dual Intent Entity Transformer (DIET) Classifier for intent classification and entity extraction. Based on that we get the intent to understand user request. The results of training are presented in figure. In



```
%[covid]('training': '12000', 'testing': '1000')
    ~[please?] ~[verb] ~[me?] ~[article?] @[topic]
    ~[verb] ~[me?] ~[article?] @[topic] ~[please?]
    ~[please?] ~[verb] ~[article?] @[topic]
    ~[verb] ~[article?] @[topic]  ~[please?]

~[verb]
    tell me
    find
    search
    search for

@[topic]
    ~[tips]
    ~[vaccine]
    ~[statistics]
    ~[report]
    ~[other]
    ~[mood]

~[tips]
    tips
    advice
    behavior information

~[report]
    news

~[mood]
    mood
    sentiment
    feeling
    thoughts
```

```
(env) kasia@Kasia:~/catkin_ws/src/slang$ python -m rasa_nlu.train -c config
/bringMe_config.yml --data data/covid_train.json -o output --fixed_model_na
me RasaCovid --project RasaCovid --verbose
/home/kasia/env/local/lib/python2.7/site-packages/rasa_nlu/utils/__init__.p
y:239: YAMLLoadWarning: calling yaml.load() without Loader=... is deprecate
d, as the default Loader is unsafe. Please read https://msg.pyyaml.org/load
 for full details.
  return yaml.load(read_file(filename, "utf-8"))
2020-11-17 15:28:03 INFO     rasa_nlu.training_data.loading  - Training dat
a format of data/covid_train.json is rasa_nlu
2020-11-17 15:28:03 INFO     rasa_nlu.training_data.training_data  - Traini
ng data stats:
        - intent examples: 8100 (2 distinct intents)
        - Found intents: 'negative', 'covid'
        - entity examples: 8098 (2 distinct entities)
        - found entities: 'topic', 'no'

2020-11-17 15:28:03 INFO     rasa_nlu.model  - Starting to train component
tokenizer_whitespace
2020-11-17 15:28:04 INFO     rasa_nlu.model  - Finished training component.
2020-11-17 15:28:04 INFO     rasa_nlu.model  - Starting to train component
ner_crf
2020-11-17 15:28:17 INFO     rasa_nlu.model  - Finished training component.
2020-11-17 15:28:17 INFO     rasa_nlu.model  - Starting to train component
ner_synonyms
2020-11-17 15:28:17 INFO     rasa_nlu.model  - Finished training component.
2020-11-17 15:28:17 INFO     rasa_nlu.model  - Starting to train component
intent_featurizer_count_vectors
2020-11-17 15:28:17 INFO     rasa_nlu.model  - Finished training component.
2020-11-17 15:28:17 INFO     rasa_nlu.model  - Starting to train component
intent_classifier_tensorflow_embedding
2020-11-17 15:28:19.021986: I tensorflow/core/platform/cpu_feature_guard.cc
:141] Your CPU supports instructions that this TensorFlow binary was not co
mpiled to use: AVX2 FMA
2020-11-17 15:28:19 INFO     rasa_nlu.classifiers.embedding_intent_classifi
er  - Accuracy is updated every 10 epochs
Epochs: 100%|██████████| 300/300 [01:38<00:00,  3.05it/s, acc=1.000, loss=0
.001]
2020-11-17 15:29:57 INFO     rasa_nlu.classifiers.embedding_intent_classifi
er  - Finished training embedding policy, loss=0.001, train accuracy=1.000
2020-11-17 15:29:57 INFO     rasa_nlu.model  - Finished training component.
2020-11-17 15:29:57 INFO     rasa_nlu.model  - Successfully saved model int
o '/home/kasia/catkin_ws/src/slang/output/RasaCovid/RasaCovid'
2020-11-17 15:29:57 INFO     __main__  - Finished training
```

Figure 9. Dataset generation and training.

Figure 9, the application workflow is presented. I shows the role of each element. The data from the first part of project was applied by hand from the results and used to give answers to the user.

6.2. Result and Future Work

The application is interesting and for a class project it is a good practice but if properly expanded could be used in real life in, for example clinics or homes to keep users inform about the pandemic. Due to the lack of access the application wasn't implemented on the HSR robot but can be easily done so in near future. I would also like to use Twitter API to get the most recent data for the application and limit hard coding to only necessary parts. This could be also expanded to other topic or infectious diseases.

6.3. Conclusions

In conclusion, the task was a good learning experience of different data mining methods. I found it useful while going through large amounts of data. If I was to complete the same task by hand, it would likely take me ten times the time than having it do by a machine. There is a lot of space for improvement and given time and resources this project could turn into a useful tool. I would definitely like to use other classification techniques and test the application on the HSR robot.

Figure 10. Word clouds.

References

[1] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[2] V. R. DEX, M. Bohanec. Car evaluation, 1990.

[3] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[4] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9(3):90–95, 2007.

[5] G. Kesavaraj and S. Sukumaran. A study on classification techniques in data mining. In *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pages 1–7, 2013.

[6] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proceedings of the First International Conference on Autonomous Agents*, AGENTS '97, page 340–347, New York, NY, USA, 1997. Association for Computing Machinery.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] S. Raschka. Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *The Journal of Open Source Software*, 3(24), Apr. 2018.

[9] G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[10] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[11] T. Yamamoto, T. Nishino, H. Kajima, M. Ohta, and K. Ikeda. Human support robot (hsr). In *ACM SIGGRAPH 2018 Emerging Technologies*, SIGGRAPH '18, New York, NY, USA, 2018. Association for Computing Machinery.