1MD032 - Intelligent Interactive Systems

Project Report

Natural Language Understanding

Students:

Abdalah Hilmia
Abdul Rahman Khankan
Mohamad Alsioufi

Abstract

The goal of this project is to build a language understanding tool for a household robot (personal assistant). The task is to implement the ability of the robot to understand 8 different commands. The final result was a system that was able to understand 12 different commands with 97% accuracy.

Acknowledgement

This report is part of a project done in Uppsala University for the course 1MD032 - Intelligent Interactive Systems. All data collected for this project was completely anonymous and used only for the purpose intended for the project for natural language understanding.

Summary

Goal

Identify 8 different tasks using voice.

Main results

Identifying 12 different tasks using voice commands, the best average accuracy was 97% using KNN & random forest classifiers.

Work schedule

The 3 of us live closely and we were either present together or on conference call when working on the project. We worked in iterations (1 week/each) and divided the work so two in pair would work together on the coding to correct each other, and the third reviewing and optimizing another aspect, we updated each other after we finished a big part and switched roles after each iteration.

Collecting Data

We collected data on 2 steps, text and voice. The dataset collected was for **12** tasks.

Text data

For this part we used Google Forms to set up a survey to collect text commands of the tasks, for each task, we showed an image or two describing the task, and asked the user to type how they would ask/order someone to perform the task illustrated by the image, we tried to use gender neutral images as well as general illustration to not specify certain part of the task

The reason why we collected text data instead of voice is because we were using a ready API for Speech-to-Text, so the accuracy was not something we would be affecting, thus, the voice data was not necessary at that point, it was also easier to start with text data and then gather the voice data later.

Survey link: http://goo.gl/forms/82SOwFwDc5o4kPWF2

Voice data

We put up together a website to collect transcribed voice dataset of our commands to do a final test on start-to-end accuracy.
We used the same images used in the survey, and asked the user to record and type how would they ask someone to do the task illustrated by the picture.

Website link: http://www.groceryassistant.co/iis/ (note that due to technical issues, the recording only works on Firefox / Edge / IE).

System description

We have logically divided the system into 3 subsystems

1. Speech-to-Text
2. Text-to-Text (Pre-processing)
3. Text-to-Class (Classification)

Speech-to-Text

This part handles the process of converting speech to written text.
There are 2 kinds of speech to text processors

- Online
- Offline

The offline processors provide an internet-independent tool with accuracy that depends on how often it gets updated, and its speed is dependent on the processing power of the system that it is being run on. However, there are some customized offline systems which run faster than online systems, but they are customized for certain commands [1]. The online processors however give higher accuracy for general recognition since the processing power and available storage for training data is not an issue anymore but they rely on internet connection.

For this part, we did not reinvent the wheel, we used an existing online speech to text API.
We evaluated 2 APIs for our project:

1. Google Speech Recognition API
2. Wit.ai API

After our evaluation (details in Results section) we used Google Speech Recognition API as it had higher accuracy.

Text-to-Text (Pre-processing)

This part handles the pre-processing step, it takes the text output from the speech API and does the following

1. Letter-wise operations
    a. Trim any extra spaces.
    b. Convert the string to lowercase.
    c. Remove punctuation.

2. Word-wise operations
    a. Remove stopwords defined by Python's Natural Language Toolkit (NLTK) English stopwords list.
    b. Remove custom stopwords that we defined by studying the dataset and its frequency analysis e.g. ['could', 'would', 'please'].
3. Natural Language Processing operations
    a. Stemming
        i. Porter Stemmer, as a way to generalize and include any variations of our training set.

Text-to-Class (Classification)

This part handles the classification of the command, it takes the processed text from the previous subsystem and outputs its class (recognized task)

It includes 2 phases

1. Training
2. Evaluating

Training

This phase is training the classifier using our training set and then evaluating each classifier model.
We started by splitting our dataset into 80% training set and 20% test set. Then, we started searching for an appropriate feature vector to be the input for the different classifiers, after some research [2][3], we decided to go with the Bag of Words or “Bag of n-grams” representation, where each sentence is represented by a vector of 0s of length n, where n is the number of distinct words in our data, and 1s in the indices that match the words of the sentence.
Afterwards, we used the data to train different classifiers:

- SVM, with different kernels
    - Linear
    - Radial Basis Function (RBF)
    - Polynomial
    - Sigmoid
- KNN, with different weights
    - Distance
    - Uniform
- Naive bayes
    - Gaussian
    - Multinomial
- Decision trees
- Random forest

The results of evaluating each classifier will be discussed in the *Results* section

Evaluating (Respond)

This phase is after the classifier is trained, it uses the user's command input to the trained classifier and outputs the recognized class.

Results

We each of the 3 subsystems separately and then did a test for the whole system using the voice transcribed dataset. The whole system gave similar results to the Text-to-Class subsystem, we believe it is because all the commands were simple and the audio was clear, thus, the Speech-to-Text parts were all recognized correctly, so the accuracy was only dependant on the text classifier.

Separate subsystem evaluation

Speech-to-Text

For this part, we evaluated the accuracy of the 2 APIs using transcribed voice dataset available for free at http://voxforge.org/, we used randomly selected people of different genders

	Google Speech Recognition	Wit.ai
Speaker 1: "besides that noise makes me deaf"	besides that noise makes me death	the sunset noise next week at
Speaker 1: "within himself he called it no longer his own"	within himself he called it no longer his own	within himself he called it no longer his own
Speaker 1: "I'll only be in the way"	I'll only be in the way	i would only be in the way
Speaker 2: "we have been chased by them ourselves more than once"	we have been chased by them ourselves more than once	we have been chased by the myself more than once
Speaker 2: "Saxon waited, for she knew a fresh idea had struck Billy"	sex and weighted fishing lure fish idea had struck Billy	section related pension of fresh idea had stuck billy
Speaker 2: "Thus was momentum gained in the Younger World"	fast food movement indeed and the younger Boyle	this was momentum gained in the underworld

We did more tests and the clear result was that Google API gave better results.
We could not find a report or paper on accuracy for speech recognition systems that we could refer to.

Text-to-Text

The only problems we came across were in the stemming process. The mistakes in stemming come from the fact that Porter stemmer is not perfect, it is a simple and very fast rule-based stemmer, it was a trade-off between accuracy and speed, but due to the logical reason that the same error will occur in both the training and in the testing, we can then neglect the error. The other steps were simple alterations in the letters and words of the string. That is why we did not do an extensive testing for this part.

Text-to-Class

The table below shows the accuracy of the different classifiers we tried on the dataset (80% training / 20% testing)

SVM, linear	0.956
SVM, RBF	0.130
SVM, Polynomial	0.130
SVM, Sigmoid	0.130
KNN, Distance	0.978
KNN, Uniform	0.934
Gaussian naive bayes	0.869
Multinomial naive bayes	0.934
Decision trees	0.913
Random forest	0.978

Conclusion, Future Work, Limitations & Ethical Considerations

The overall accuracy of the system was better than we had expected before we started, however, we think there's still room for improvement. We looked at other stemmers, Porter2(Snowball) and Lancaster, however, even though Porter2 has a slight improvement over Porter in terms of accuracy and speed, we used Porter because it was the simplest and it served our needs. We believe a better stemmer could match more sentences with our model. We also thought about applying Parts-of-Speech tagging and semantic modeling (HMM), which we think in theory it will make the system understand more complex commands.

The main limitation was the dataset, in both

1. Size: the survey had only 17 responses, and only 3 participated for the recorded dataset which causes inconclusive results, we do not know how the system would perform on larger dataset.
2. Content: we had some sentences like: "cook some vegetables i'm vegan don't put much attention on whether the pan overflows" which are too complicated, some other sentences like: "flowers are looking dead again" or "be a chef" which are ambiguous, some like "can you help me with this" which are confusing even for a human, finally, sentences like "do the laundry also smile like your life is not miserable" which were an attempt of joke.

We think the experiment should be evaluated again with a better dataset.
Also, we should keep in mind ethical consideration, one problem if we were training our Speech-to-Text is a "Racist Classifier" when it is trained using people with specific accents. We should also address potential security issues when the robot will be idle, will it be listening 24/7 ? If the Speech-to-Text is online, then that data be sent for analysis, will it be stored on a 3rd party's server? What are the 3rd party's policy regarding the handling of that data? Some speech recognition companies share and sell their data since it is anonymous, but what if it contained sensitive information? Like when the robot is listening and you are telling someone (your spouse) your credit card information, it is anonymous data, but still very sensitive.
All those ethical concerns must be addressed when implementing such a system that has access to potentially very sensitive data.

References

[1] McGraw I, Prabhavalkar R, Alvarez R, Arenas MG, Rao K, Rybach D, Alsharif O, Sak H, Gruenstein A, Beaufays F, Parada C. Personalized Speech recognition on mobile devices. arXiv preprint arXiv:1603.03185. 2016 Mar 10.

[2] Srivastava AN, Sahami M, editors. Text mining: Classification, clustering, and applications. CRC Press; 2009 Jun 15.

[3] http://scikit-learn.org/stable/modules/feature_extraction.html

Appendix A: Original specification

Background:
In social robotics we expect robots to communicate with us the same way other people do. Natural spoken language is one of the key communication abilities of humans, so we would like robots to understand normal spoken language and process it semantically to make sense of it.

Task:
You are building the language understanding tool for a household robot. The task is to implement the ability of the robot to **understand 8 different commands**, e.g. “Clean the dishes”, “Do the laundry”, “Tidy up the flat”, but because **different people** might use the robot, the exact wording of the command can be different and need to be learned from human test users.

Project plan & timeline:

Deadline: **Task**

- 28/04: Prepare a survey for potential users to define commands with corresponding tasks.
- 29/04: Publish and spread the survey.
- 02/05: Start preparing dataset according to the survey results.
- 03/05: Research machine learning techniques used in the field and select the most suitable one for natural language understanding.
- 06/05: Build the machine learning model.
- 10/05: Use the data as input for the machine learning model.
- 10/05: Evaluate the application results using an appropriate evaluation method.
- 11/05 - > 25/05: Optimize the machine learning model and evaluate new results.
- 28/05: Report final results.
- 29/05: Make a demo.
- 29/05: Comment any unclear parts of the source code.
- 29/05: Reflect and elaborate on potential ethical concerns identified with the system.
- 30/05: Wrap-up and finish the report.
- 31/05: Submit the report.