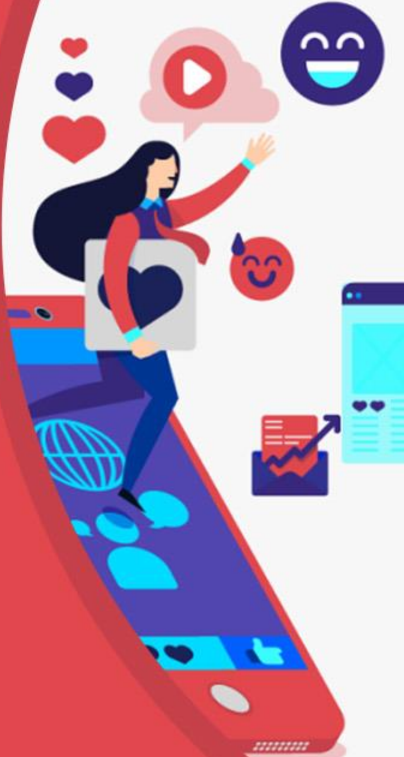
CHATBOT FOR INTENT RECOGNITION

Presented by
Adina Dingankar
Rehapriadarsini Manikandasamy
Venkata Gangadhar Naveen Palaka

Group 6 – DATS 6312 Natural Language Processing



INTENT RECOGNITION (IR)

- Intent recognition is sometimes called as intent classification which is the task of taking a written or spoken input, and classifying it based on what the user wants to achieve
- Intent recognition works through the process of providing examples of text alongside their intents to a machine learning (ML) model

APPLICATIONS: Chatbots, customer support, sales prospecting, etc.,

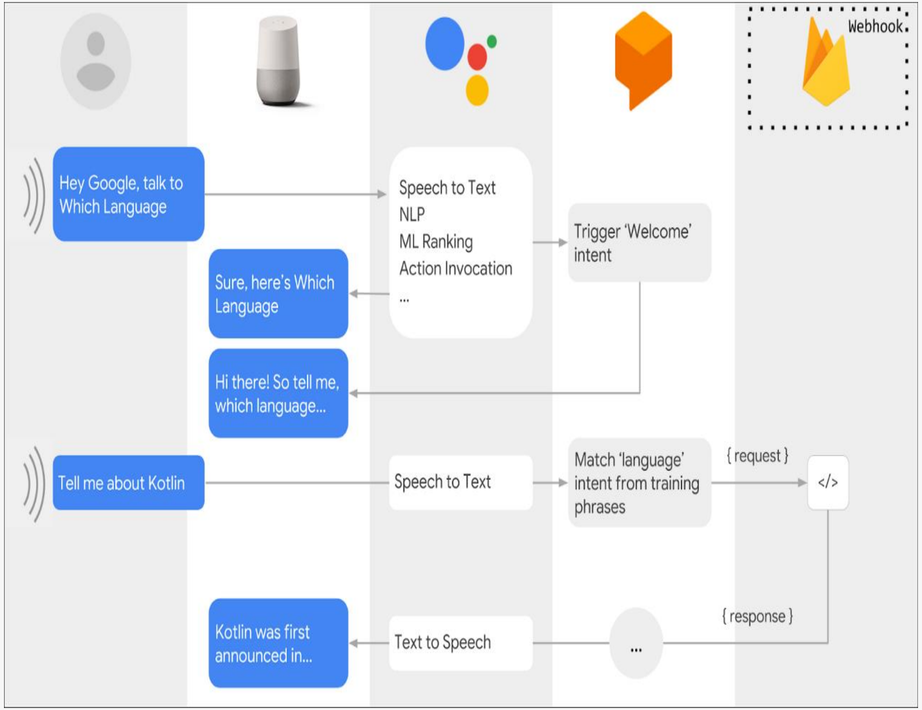
CHATBOTS USE IR?

- Intent recognition is a critical feature in chatbot architecture that determines user's aim in starting any conversation
- NLP allows the chatbot to understand the user's message, and machine learning classification algorithms to classify the message based on the training data and deliver the correct response
- The chatbots' intent detection component helps to identify what general task or goal the user is trying to accomplish to handle the conversation with different strategies

GOOGLE & IR

- Google's Dialogflow - Platform to design and integrate a conversational user interface
- When an end-user writes or says something, referred to as an end-user expression, Dialogflow matches the end-user expression to the best intent in your agent

Used by : Dominos, Malaysian Airlines, KLM Royal Dutch Airlines, Verizon, CNN, etc.,



DATASET

- Motivation – NLU Benchmark (Few-Shot-Detection)
- Few-Shot-Intent-Detection is a repository designed for few-shot intent detection with/without Out-of-Scope (OOS) intents

Intents

out-of-domain OOS
General out-of-scope queries which are not supported by the dialog systems

in-domain OOS
Queries which are more related to the in-scope intents, which makes the intent detection task more challenging

DATASET

- The scope of our dataset has been inherited from ‘CLINC150’ intent dataset
- Our purpose was to built a chatbot for intent recognition we have developed a custom-built dataset

Features of Dataset:

Intents
User input
Bot responses

```
{"intents":
[{"tag": "greeting",
   "patterns": ["Hi", "How are you", "Is anyone there?", "Hello", "Good day"],
   "responses": ["Hello, thanks for visiting", "Good to see you again", "Hi there, how can I
help?"],
]}}
```

MODELS

- BERT
- DISTIL BERT
- LSTM

WHAT IS THE NEED FOR BERT?

- ❑ BERT uses bidirectional training
- ❑ It takes both the previous and next tokens into account simultaneously
- ❑ BERT applies the bidirectional training of Transformer to language modeling, learns the text representations

BERT

- Packages : Bert-for-tf2, BertModelLayer
- Pretrained model : uncased-bert-pretrained-model
- Tokenizer : FullTokenizer
- The pretrained bert model is embedded to the keras model with two dense layers and two dropout layers with value of 0.5 using the activation function of tanh and softmax

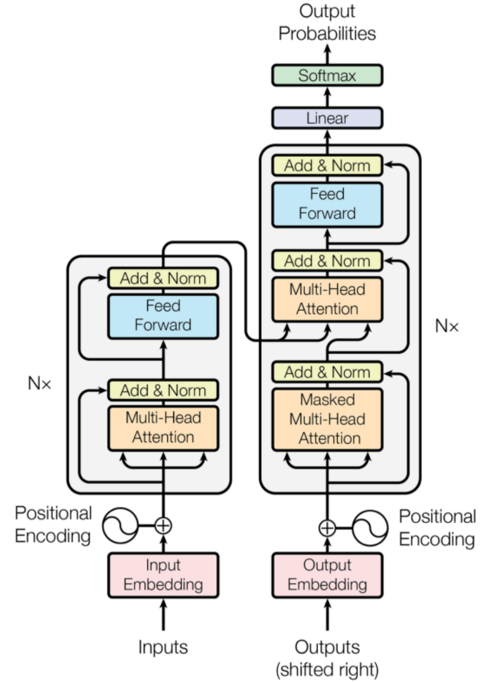


Figure 1: The Transformer - model architecture.

BERT

```
model.summary()

Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_ids (InputLayer)       [(None, 16)]              0
_________________________________________________________________
bert (BertModelLayer)        (None, 16, 768)           108890112
_________________________________________________________________
lambda (Lambda)              (None, 768)               0
_________________________________________________________________
dropout (Dropout)            (None, 768)               0
_________________________________________________________________
dense (Dense)                (None, 768)               590592
_________________________________________________________________
dropout_1 (Dropout)          (None, 768)               0
_________________________________________________________________
dense_1 (Dense)              (None, 117)               89973
=================================================================
Total params: 109,570,677
Trainable params: 109,570,677
Non-trainable params: 0
_________________________________________________________________
```

```
print("train acc", train_acc)
print("test acc", test_acc)

train acc [4.759221453693032, 0.07734807]
test acc [4.759221453693032, 0.07734807]
```

Accuracy is very poor

Reason: The training and test data of minimum 8000 entries respectively is required which was major drawback of our use case since it didn't had test set independently and was populated using train set itself

DISTILBERT

- Proposed to pre-train a smaller general-purpose language representation model
- It is the size of a BERT model reduced by 40%, while retaining 97% of its language understanding capabilities and being 60% faster
- Distil BERT is smaller, faster and lighter model is cheaper to pre-train and demonstrates its capabilities for on-device computations in a proof-of-concept experiment and a comparative on-device study

DISTILBERT

	BERT	RoBERTa	DistilBERT	XLNet
Size (millions)	**Base**: 110 **Large**: 340	**Base**: 110 **Large**: 340	**Base:** 66	**Base**: ~110 **Large**: ~340
Training Time	**Base**: 8 x V100 x 12 days* **Large:** 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)	**Large:** 1024 x V100 x 1 day; 4-5 times more than BERT.	**Base**: 8 x V100 x 3.5 days; 4 times less than BERT.	**Large:** 512 TPU Chips x 2.5 days; 5 times more than BERT.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT	3% degradation from BERT	2-15% improvement over BERT
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)	16 GB BERT data. 3.3 Billion words.	**Base**: 16 GB BERT data **Large**: 113 GB (16 GB BERT data + 97 GB additional). 33 Billion words.
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**	BERT Distillation	Bidirectional Transformer with Permutation based modeling

DISTILBERT

- For Distil BERT implementation we are making use of distil-bert-uncased model
- The data is tokenized by using the pattern column of the data, and building a tensor out of the padded input and sending it to the distil bert model
- The model's performance is measured by using the classification report from sklearn.metric package.

accuracy			0.32	37
macro avg	0.11	0.14	0.12	37
weighted avg	0.26	0.32	0.29	37

LSTM

- Introduced to avoid the long-term dependency problem
- LSTMs efficiently improve performance by memorizing the relevant information that is important and finding the pattern
- In LSTM we can use a multiple word string to find out the class to which it belongs
- Use of appropriate layers of embedding and encoding in LSTM, will be able to find out the actual meaning in input string and will give the most accurate output class

GLOVE

- It is an unsupervised learning algorithm developed by researchers at Stanford University aiming to generate word embeddings by aggregating global word co-occurrence matrices from a given corpus
- The basic idea behind the Glove word embedding is to derive the relationship between the words from statistics

LSTM

- Pretrained LSTM model is used
- GloVe : 6B.50d and 100d .txt
- Performed LSTM model building by embedding 1 dense layer with SoftMax activation function

LSTM

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 10)]              0
_________________________________________________________________
embedding_1 (Embedding)      (None, 10, 50)            20000850
_________________________________________________________________
lstm_1 (LSTM)                (None, 10, 128)           91648
_________________________________________________________________
dropout_1 (Dropout)          (None, 10, 128)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               131584
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 42)                5418
_________________________________________________________________
activation_1 (Activation)    (None, 42)                0
=================================================================
Total params: 20,228,700
Trainable params: 228,650
Non-trainable params: 20,000,050
_________________________________________________________________
Model: "model"
```

```
11/11 [==============================] - 1s 6ms/step - loss: 0.2037 - accuracy: 0.9014
loss: 0.2037387639284134 Accuracy: 0.9014492630958557
```

GloVe: 6B.50d
Accuracy : 0.90
Loss: 0.203

GloVe: 100d
Accuracy : 0.89
Loss: 0.2

WINNER

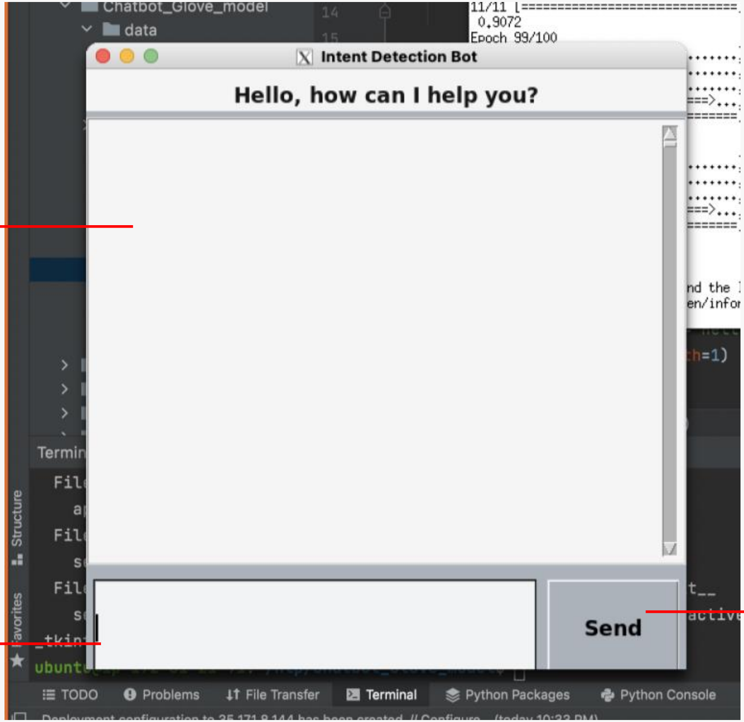
LSTM	0.904
BERT	0.077
DISTILBERT	0.32

GUI

- Package : Tkinter
- To run the GUI, Xquartz has been used
- The GUI interacts with the user and retrieves the response using the best model **'LSTM'**

RESULTS

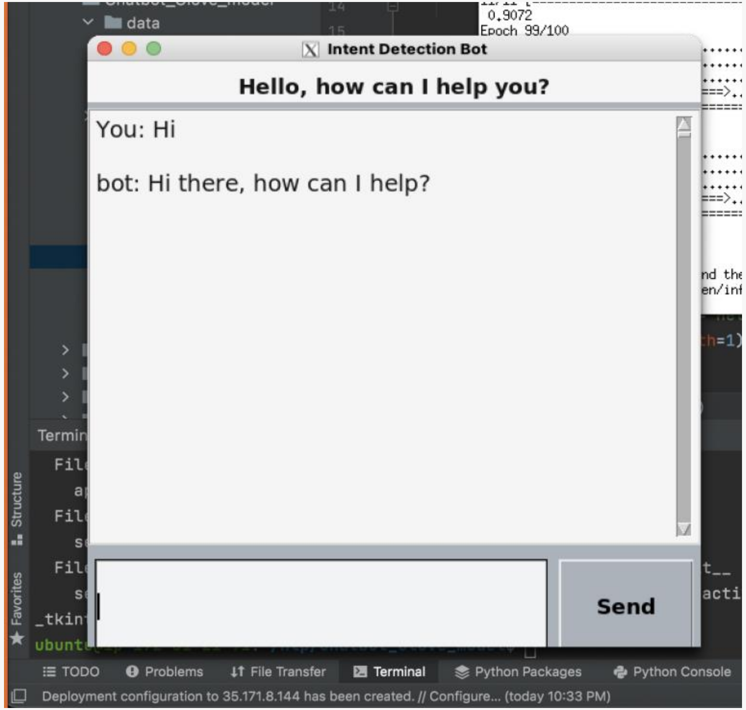
Conversation logs

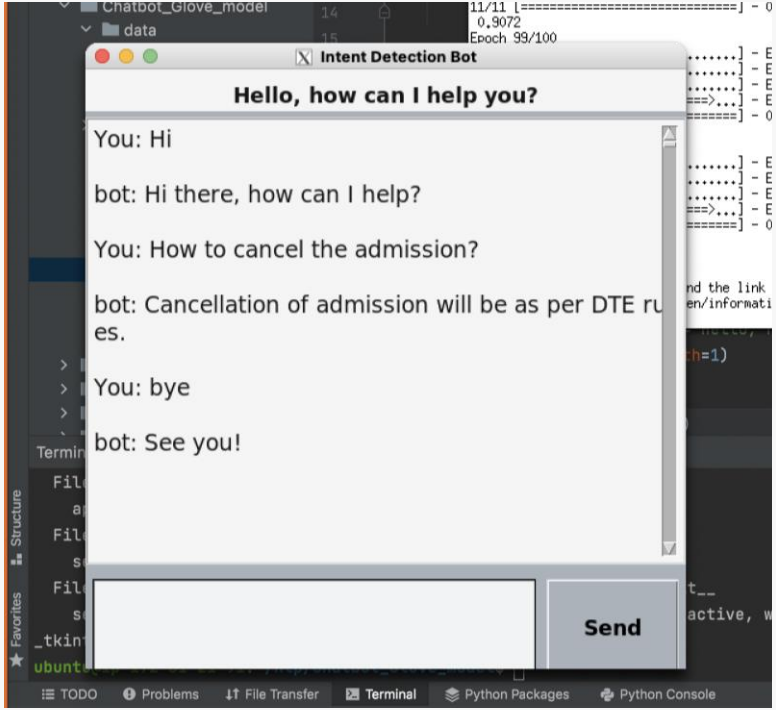


Type your query

Press to send a message to the bot

RESULTS


Intent Detection Bot
Hello, how can I help you?
You: Hi
bot: Hi there, how can I help?
Send
TODO
Problems
File Transfer
Terminal
Python Packages
Python Console
Deployment configuration to 35.171.8.144 has been created. // Configure... (today 10:33 PM)


Intent Detection Bot
Hello, how can I help you?
You: Hi
bot: Hi there, how can I help?
You: How to cancel the admission?
bot: Cancellation of admission will be as per DTE ru
es.
You: bye
bot: See you!
Send
TODO
Problems
File Transfer
Terminal
Python Packages
Python Console

CONCLUSION & FUTURE SCOPE

- After making a comparison of all the model implementation we got to observe that LSTM is having the highest accuracy of 90%
- The further enhancements which can be done to this data is to increase the size of the train set and populate a test data
- Implementation of ConveRT - A dual sentence encoder , it is effective, affordable, and quick to train also the size of the ConveRT model is less compared to the BERT model

REFERENCES

- https://arxiv.org/abs/1805.10190
- https://github.com/sonos/nlu-benchmark/tree/master/2017-06-custom-intent-engines
- https://github.com/huggingface/transformers
- https://paperswithcode.com/task/intent-detection
- https://www.sciencedirect.com/science/article/pii/S1877050918320374
- https://analyticsindiamag.com/hands-on-guide-to-word-embeddings-using-glove/

PROJECT REPO

- The project implementation and codes can be found on the following repo:

https://github.com/Rehamanikandan/Final-Project-Group6