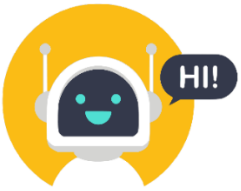


NEO

AI Conversational Agent

FYP Team

Khizar Shabir 16I-0294

Abdul Mogees 16I-0199

Supervised By: Dr. Mirza Omer Beg

Co-Supervisor: Mr. Shoaib Mehboob

FAST School of Computing

National University of Computer and Emerging Sciences

Islamabad, Pakistan

2019

ANTI-PLAGIARISM DECLARATION

This is to declare that the above publication produced under the:

Title: Neo – AI Conversational Agent

is the sole contribution of the authors and no part hereof has been reproduced on as it is basis (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. The work presented in the report is our own and it has not been submitted/presented previously to any other institution or organization. We will be responsible and liable for any consequence if violation of this declaration is determined.

Date: 01-08-2019

Student 1

Name: Khizar Shabir

Signature:

Student 2

Name: Abdul Mogees

Signature:

Supervisor

Name: Dr. Omer Beg

Signature:

Co-Supervisor:

Name: Mr. Shoaib Mehboob

Signature:

Contents

Chapter 1. Introduction

1.1 Problem Domain

Since the beginning of the field of computer science, man has always been fascinated by the idea of a machine that is just like humans. This fascination was the main drive behind the emergence of new fields like Artificial Intelligence. Since the mid of 20th century, scientists are working to convert this fascination into reality. This struggle goes from ELIZA, the first AI chatbot, all the way to MITSUKU, four times winner of Loebner Prize. Work in this field has always been more focused on a close domain. Building a conversational agent that can chat on various topics is a very challenging task. The goal of an open-domain dialog agent is to maximize the long-term user engagement. One big challenge with building these kind of conversational agents is the consistency in their conversation. Conversational consistency is very important if a conversational agent wants to build trust and long term confidence.

Over the next few months, we intend to build a conversation expert with personality. It will be able to chat on various topics, with its conversation consistent to its persona. Agent that can chat with humans in the way that people talk to each other will be easier and more enjoyable to use in our day-to-day lives — going beyond simple tasks like playing a song or booking an appointment.

In the past, absence of big conversation datasets was one of the main reason of the scarcity of conversational agents and their poor performance. But now big datasets are available by using which agent's conversation can be significantly improved. Main challenges in this project include, understanding the context of the conversation, building persona of the agent and improving the consistency of the agent with respect to its persona.

1.2 Research Problem Statement

Absence of consistency in a conversational agent has been a long standing problem. Conversation consistency is very important if a conversational agent wants to build trust and long term confidence. Personality in a key differentiator in a conversational agent. We will be working on improving the conversation consistency of the agent according to its persona.

Chapter 2. Literature Review

2.1 Improvised Theatre alongside Artificial Intelligences by Kory W. Mathewson, Piotr Mirowski, 2017 [2]

Summary

In this paper authors present the first report of Artificial Improvisation, or improvisational theatre performed live, on-stage, alongside an artificial intelligence-based improvisational performer. Authors discuss that recent work in conversational agents has been focused on goal-directed dialogue focused on closed domains such as appointment setting, bank information requests, and question-answering. Natural human conversations are seldom limited in scope and jump from topic to topic. Taking this idea in consideration, they have built two artificial improvisors, **1: Pyggy** using classic machine learning and deterministic rules to Version **2: A.L.Ex.** which uses deep neural networks, advanced natural language processing, and a much larger training dataset.

Pyggy's Dialogue management was handled with Pandorabots and Chatterbox. Pandorabots handled hard-coded rules and deterministic responses. For example, when the human said: "Let's start improvising", the system would always respond: "Ok". Chatterbot was introduced to handle open dialogue and add a sense of randomness to the system. Pre-training of Pyggy was done through an interactive website where individuals could directly interact in basic chit-chat. Additional clean training data was appended from the Cornell Movie Dialogue Corpus. The dataset is composed of 220579 conversational exchanges from 617 movies. There were limitations to the dialogue which Pyggy could produce, as it was restricted to the set of sentences present in the training data. Pyggy had no means by which to understand or track the topic of a scene. These limitations prompted them to explore a word-by-word generation approach.

They decided to imitate the creative process of improvisation using a statistical language model that can generate text as a sequence of words. This new model called A.L.Ex

(Artificial Language Experiment) was built using recurrent neural networks (RNN) with long-short term memory (LSTM). They experimented with multiple LSTM architectures with the goal of building a dialogue model that can handle the topics with in an improvised scene over dozens of exchanges between the human and the AI. They used a Latent Dirichlet Allocation topic model that enables the language model to integrate long-range dependencies in the generated text and capture the general theme of the dialogue. The language model of A.L.Ex. was trained on transcribed subtitles from 102,916 movies from OpenSubtitles.org, going from 1902 to early 2016.

Critical Analysis

A.L.Ex. was designed to subvert the multiplicity of connected services which formed the architecture of Pyggy. A.L.Ex. has been performed 24 times and on two continents. The dialogue system somewhat manages to keep track of the general theme of the conversation. Dataset used to train this model is its biggest strength, which is very is very diverse.

Relationship with Work

Our project is mainly inspired by this paper. This paper is crucial for our approach formulation as the methodologies used in the paper brought substantial improvement in the conversation of the agent. We intend to build a neural generative based model for dialog generation. We will incorporate an architecture similar to the architecture of A.L.Ex. built with Latent Dirichlet Allocation topic model in our language model.

2.2 A Neural Chatbot with Personality by Huyen Nguyen, David Morales, Tessera Chin, 2017 [3]

Summary

Authors of this paper discuss some of the issues that one has to face when working with sequence to sequence models for building generative chatbots. In 2014, these models opened some new possibilities of phrasing dialogues as a translation problem: translating from an utterance to its response. Systems having this principle as their backbone were considered good at fluent conversations but they were not convincing due to lack of

personality and inconsistent persona. In this paper, they experiment building open-domain response generator with personality and identity. They built chatbots that imitate characters in popular TV shows: Barney from How I Met Your Mother, Sheldon from The Big Bang Theory, Michael from The Office, and Joey from Friends.

Conversational modeling can be phrased as a mapping between utterances and responses, and therefore can benefit from the encoder-decoder setup of sequence models. In their model, the encoder processes an utterance by human, and the decoder produces the response to that utterance.

Initially, they used the Cornell Movie-Dialogs Corpus, which is a well formatted dataset of dialogues from movies. While it is well formatted, this dataset has a huge problem which is the conversations are very dramatic and unrealistic. The chatbot built by using this dataset seemed fake and over-dramatic. Moreover, erratic changes in movie situations created inconsistencies in the chatbot. So, they decided to use dialogue from TV shows whose content more closely modeled reality and whose characters spoke more like average people. Considering that each scene has its own topic, context, and speakers and addressees, they split the dialog scripts into conversations of contiguous utterances.

The model has a close resemblance to Google's Neural Machine Translation model published in 2016. It's a sequence-to-sequence model with attentional mechanism. Due to small amount of utterances for each character that they train on, they first trained these bots on more general datasets.

Their training involved 3 phases, each involving a different set of data and a different number of training iterations. In phase 1, they trained on all five datasets, the four TV show transcripts and the Cornell Movie-Dialogs Corpus, for capturing the basic dialog patterns. In phase 2 they limited the training to only four TV show scripts for tuning the model to be more realistic. In phase 3, they finish by training their character chatbots on only the utterances made by the characters they are trying to emulate. This final phase fine tunes the model to be more specific to a particular character. They only used character token id for encoders, not decoders, so the responder doesn't know who she/she is talking to.

To test their models, they use both automatic metrics and human judgment. For automatic metrics, they used the BLEU and ROUGE metrics. The BLEU metric uses a modified n-gram precision score that attempts to model a human judgment of how accurate a

translation is. This automated evaluation of their chatbots produce very low scores. They felt that these scores are not fair because in case of conversation, there are often infinitely many responses that would be appropriate, even within the constraints of one personality. So they created a human evaluation of chatbots which show that for a bot's response, a human is more than 50% likely to believe that the response actually came from the real character.

Critical Analysis

The scripts that they created suffered from inconsistencies like missing punctuation, grammatical errors, spelling errors that can affect model's learning. Another problem with their scraped dataset was that each TV shows has at most 50,000 pairs of utterance-response, which is not enough information to train a neural net chatbot on. The characters also seem to learn their identities. If they ask "who are you"? each character answers this question in a distinct way. This is quite remarkable since none of these exact utterance-response pairs are in the training sets. Better results can be achieved if we use a bigger and diverse dataset.

Relationship with Work

This model demonstrates that it's possible to encode certain aspects of the personality and identity of a character in the chatbot. Using the generic datasets to train phase 1 and phase 2, we can use train phase 3 on a much smaller dataset to make the bot talk like a certain character. This type of training can help a lot with for maintaining the persona consistency of Neo. By using that technique, we will encode the personality of Neo.

2.3 Dialogue Natural Language Inference by Sean Welleck, Jason Weston, Arthur Szlam, Kyunghyun Cho, 2019 [4]

Summary

In this paper authors present a new approach to increase the consistency of any conversational agent. Lack of consistency is still a big issue in all the existing chatbots.

Approach discussed in this research paper is that each agent is associated with a persona which is a set of lines defining personality of that agent. These researchers used **NLI Dataset** from Facebook that contains pairs of sentences from persona and utterances sets. Now each persona sentence is associated with a human-labelled *triple* (e_1, r, e_2). Triple is <category> <relation> <category> which shows the basic theme of sentence. This task was done using Amazon Mechanical Turk service. Now each utterance from the agent is also associated with a triple and utterance is compared with the persona triple and it entitles it with one of three labels that are **Entailment, Neutral** and **Contradiction.**

When this model is combined with the existing dialogue generating models, it increases their consistency. The process is that, dialogue generating model produces a set of candidate utterances that can be the answer to the user. These candidate utterances are passed to the NLI model which ranks them on the bases of their score of Entailment, Neutral and Contradiction. The utterance that has maximum score, qualifies to be the answer that is most consistent to its persona. Moreover, in this paper a new for measuring the consistency is proposed which is effective as visible from the results shown in research paper.

	Haves		Likes		Attributes	
	Orig.	**Rerank**	**Orig.**	**Rerank**	**Orig.**	**Rerank**
Hits@1 ↑	30.2	**37.3**	16.9	**18.7**	35.2	**36.4**
Contradict@1 ↓	32.5	**8.96**	17.6	**4.1**	8.0	**5.7**
Entail@1 ↑	55.2	**74.6**	77.9	**90.6**	87.5	**88.6**

In this paper, researchers demonstrated that natural language inference can be used to improve performance on a downstream dialogue task. To do so, they created a new dialogue-derived dataset called Dialogue NLI, a re-ranking method for incorporating a Dialogue NLI model into a dialogue task, and an evaluation set which measures a model's persona consistency.

Critical Analysis

Work discussed in this paper is latest advancement in field of improving consistency in chatbots. This work is done by AI researchers of Facebook. In this paper consistency between utterances generated by chatbot and the persona of that agent is discussed. Though it is a good point but the thing that is left untouched in this research paper is the consistency between new utterances and the previous ones. The dataset for that is also available and it can also be incorporated by making slight changes. Another strength of this research paper is introducing its own metrics for measuring the consistency of a chatbot.

Relationship with Work

This paper is most related to our proposed work of improving the consistency of chatbot with its persona. Our model will be producing multiple utterances from which we will be selecting the most consistent utterance based on some model like NLI.

2.4 Context Dependent Recurrent Neural Network Language Model by Tomas Mikolov, Geoffrey Zweig, 2012 [1]

Summary

This research paper deals with Recurrent Neural Network Language Models **(RNNLM)**, having the potential to model long span context dependencies. The simple recurrent neural network language model consists of an input layer, a hidden layer with recurrent connections that propagate time-delayed signals, and an output layer, plus the corresponding weight matrices. The input vector w(t) represents input word at time t encoded using 1-of-N coding (also called one-hot coding), and the output layer produces a probability distribution over words. The hidden layer maintains a representation of the sentence history. In this research paper, they extended this basic model with an additional *feature layer* f(t) that is connected to both the hidden and output layers. The feature layer represents an external input vector that should contain complementary information to the

input word vector w(t). In the rest of this paper, they used features that represent topic information. There are several possible advantages to using topic information as additional input to the model, instead of building many separate topic-specific sub models: mainly, the training data will be less fragmented. Also, by providing the topic information directly at the input of the model, we elegantly avoid the long-standing problem of training recurrent neural networks to remember longer-term information.

Critical Analysis

In this paper, researchers presented the utilization of context dependent RNN language models. Main idea behind this is to condition the hidden and output vectors on a continuous space representation of the previous words and sentences. Along with this, the researchers developed a new fast-approximate context-updating strategy which enable us to compute context vectors more efficiently using a sliding window. These models were then used in the lowest published perplexity on the Penn Treebank data, and in WER improvements for the Wall Street Journal task. In coming future, the researchers intend to apply this particular approach to leverage external information that is not present in the text. For instance, in case of machine translation, using a source-language representation to condition the target-side language model, or in case of voice-search setting to use a contextual vector representing a user's interests.

Relationship with Work

Since our project deals with the long-term context dependencies, so this research paper is very much related to our field of work. Since we need to improve the consistency of our chatbot, so we must consider topic modeling for this. This paper is about extracting topics from conversation and this can help in improving the consistency of conversational agent.

2.5 Rasa: Open Source Language Understanding and Dialogue Management by Tom Bocklisch, Joey Faulkner, Nick Pawlowski, Alan Nichol, 2017 [5]

Summary

In this paper authors present a pair of tools, Rasa NLU and Rasa Core, which are open source python libraries for building conversational software. Before Rasa there was no widely-used statistical dialogue system intended for non-specialists. Purpose of these tools is to make machine-learning based dialogue management and language understanding accessible to non-specialist software developers. Rasa is already used by thousands of developers worldwide. As with many other conversational systems, their tools are split into natural language understanding (Rasa NLU) and dialogue management (Rasa Core).

Rasa's architecture is modular by design. This allows easy integration with other systems. For example, Rasa Core can be used as a dialogue manager in conjunction with NLU services other than Rasa NLU. Dialogue state is saved in a tracker object. There is one tracker object per conversation session, and this is the only stateful component in the system. A tracker stores slots, as well as a log of all the events that led to that state and have occurred within a conversation. The state of a conversation can be reconstructed by replaying all of the events. When a user message is received Rasa takes a set of steps as described in figure1. Step1 is performed by Rasa NLU, all subsequent steps are handled by Rasa Core.

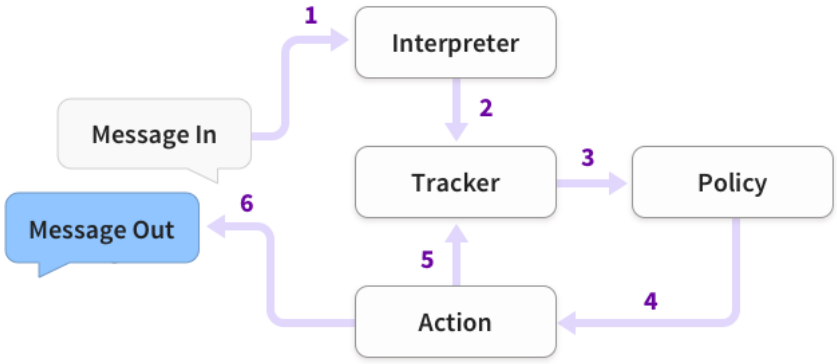


Figure 1: 1. A message is received and passed to an Interpreter (e.g. Rasa NLU) to extract the intent, entities, and any other structured information. 2. The Tracker maintains conversation state. It receives a notification that a new message has been received. 3. The policy receives the current state of the tracker. 4. The policy chooses which action to take next. 5. The chosen action is logged by the tracker. 6. The action is executed (this may include sending a message to the user). 7. If the predicted action is not 'listen', go back to step 3.

They frame the problem of dialogue management as a classification problem. At each iteration, Rasa Core predicts which **action** to take from a predefined list. An **action** can be a simple utterance, i.e. sending a message to the user, or it can be an arbitrary function to execute. When an action is executed, it is passed a **tracker** instance, and so can make use of any relevant information collected over the history of the dialogue: slots, previous utterances, and the results of previous actions. Actions cannot directly mutate the tracker, but when executed may return a list of events. The tracker consumes these events to update its state.

The natural language understanding (**NLU**), term is understood as converting short user messages into dialogue acts comprising an intent and a set of entities. Rasa NLU is the natural language understanding module. It comprises loosely coupled modules combining a number of natural language processing and machine learning libraries in a consistent API. They aim for a balance between customizability and ease of use.

The job of a **policy** is to select the next action to execute given the tracker object. A policy is instantiated along with a featurizer, which creates a vector representation of the current dialogue state given the tracker. The standard featurizer concatenates features describing: what the last action was, the intent and entities in the most recent user message, which slots are currently defined.

Both Rasa NLU and Core work with human-readable training data formats. Rasa NLU requires a list of utterances annotated with intents and entities. These can be specified either as a json structure or in markdown format. Rasa Core employs markdown to specify training dialogues (aka '**stories**').In addition to supervised learning, Rasa Core supports a machine teaching approach where developers correct actions made by the system. Rasa Core also has the capability to visualize a graph of training dialogues.

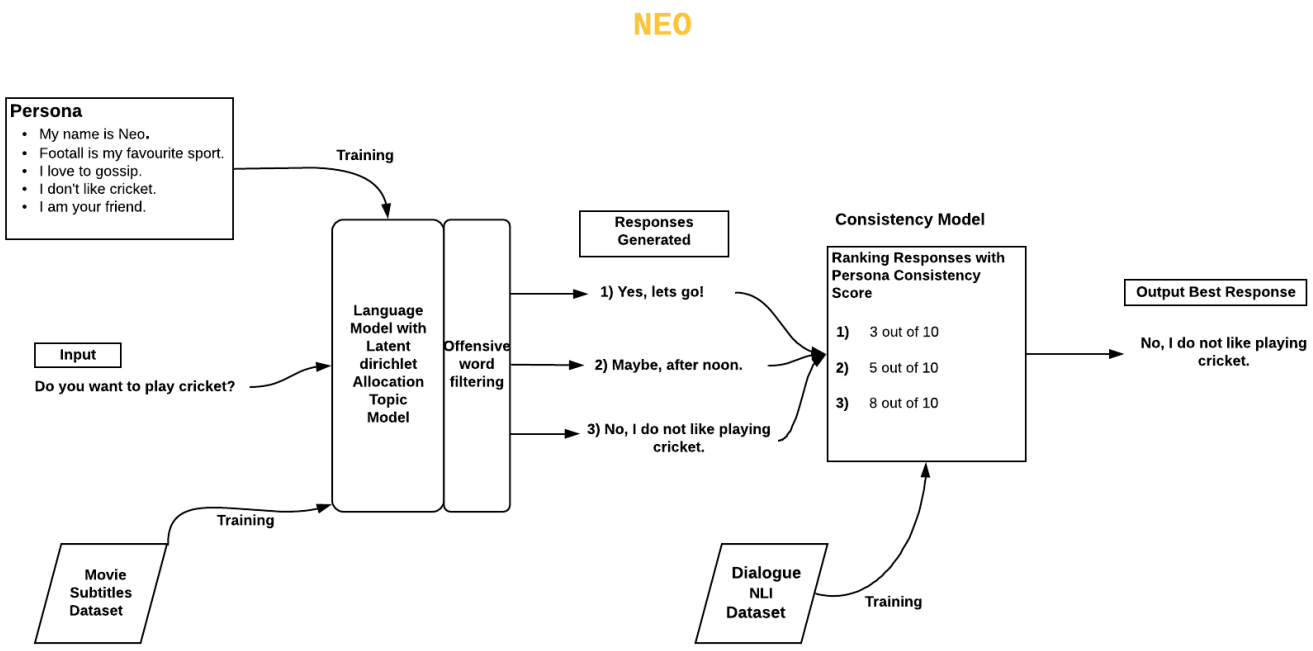
Critical Analysis

Rasa's language understanding and dialogue management are fully decoupled. This independence of both understanding and dialogue management from one another enables us to use RASA NLU and Core independently and allows trained dialogue models to be reused across different languages. Language understanding is performed by a number of components implementing a common API, and is therefore easily configurable to suit the needs of a particular project.

Relationship with Work

We will be using Rasa framework because of its modularity and customizability. Instead of using Rasa NLU, we will be using topic modeling for capturing the theme of dialogue.

Chapter 3. System Diagram



Chapter 4. Proposed Approach

There are two models that we will be building. First is language model for dialogue generation. The second model is consistency assurance model.

Language model will be a Sequence To Sequence model built by using neural network with LSTM. We can also use transformers for building this model because they have shown quite impressive results for NLP tasks in recent years. We will be training the language model on the movie subtitles data set, which contain more than 100800 subtitle files scraped from OpenSubtitles.com. We will be doing data processing, for cleaning that data. Understanding the context of conversation is very important for an agent. We will be using Latent Dirichlet Allocation topic model with the language model for that purpose. Language model will generate text word by word. Since offensive word are used very often in the movies. We will be doing offensive words filtering for that purpose. Language model will generate multiple responses for an utterance, which will later be used in consistency model.

Consistency model will ensure the consistency of agent's conversation according to its persona. It will rank all the responses generated by the language model and give them a score according to their resemblance with the agent's persona. The most suitable response with maximum score will be chosen as the ultimate response. We will be using Dialogue NLI dataset for the training of consistency model. It's approximately 400MB. There are three classes in this dataset, entailment, contradiction and neutral.

We will be deploying the final system on an online website.

Chapter 5. Dataset Preparation

A generative language model that should be able to answer questions from a vast domain needs to be trained on a large dataset that covers a vast variety of communication aspects. For this one of the best options available was the dataset of movies subtitles. This dataset was used by Kory W. Mathewson in implementing an improvisor robot.

5.1 Data Scraping

For gathering the dataset of movie subtitles, we did web scraping using R language scripts. Using this technique, we gathered a large dataset of around 10GB that comprises of subtitles of English movies from 1950s to 2015. Data downloaded this way is in form of XML files.

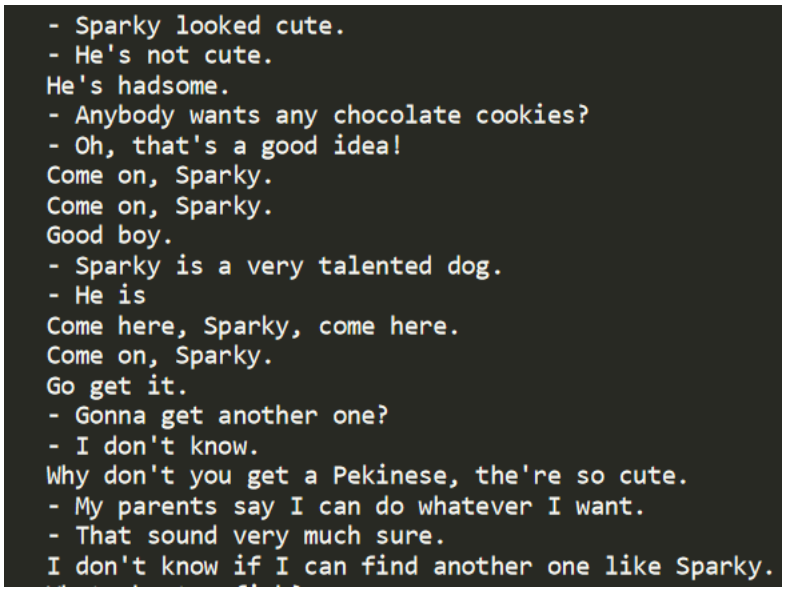
5.2 Data Format

Subtitles are saved in XML files. Movie dialogues are bounded by time tags, which gives information about starting and ending time of that particular sentence on screen. We don't need any other kind of data other than the movie dialogues. This is a sample snippet of one of the XML files,

```
  <s id="26">
    <time id="T17S" value="00:02:43,789" />
Dr. Locke, the headquarters of the network link One.
    <time id="T17E" value="00:02:48,920" />
  </s>
  <s id="27">
    <time id="T18S" value="00:02:56,218" />
Be prepared.
    <time id="T18E" value="00:02:58,266" />
  </s>
  <s id="28">
    <time id="T19S" value="00:03:00,430" />
- We know the level 1 disturbance.
  </s>
  <s id="29">
-A Minor setback.
    <time id="T19E" value="00:03:05,607" />
  </s>
  <s id="30">
    <time id="T20S" value="00:03:05,602" />
A potential disaster.
    <time id="T20E" value="00:03:07,729" />
  </s>
```

5.3 Parsing XML

We only needed the dialogue portion from these XML files. So, for that we used XML ElementTree to parse the XML files and extract only useful data. For this purpose, python script was used to parse XML files and convert them into txt files that contains only movie dialogues. Moreover, after extracting data from XMLs, there was still need to clean the data. As subtitles doesn't only contain movie dialogues but also contain sentences that explain the scene or some pre context. These sentences are mostly enclosed in some kind of bracket or some other special character. We used python regex to clean our data from this kind of gibberish data for us. After parsing and cleansing the data, final outcome looked like this,



5.4 Making Dialogues

Since our model is a dialogue generating model, so it should be trained on responses to sentences. For this we needed to convert our dataset from simple txt files into dialogue from of data, where there is a response against each sentence. Let's suppose that our txt file has 5 sentences delivered by two characters of a movie. So, the second sentence is response of first sentence and third sentence is response of second sentence and so on. This format of data will help model to learn how to generate responses for different types of sentences and questions to keep the communication going.

Chapter 6. Implementation

There are two main modules of our system. First is the language model for dialogue generation that we have implemented in this iteration and the second one is the consistency assurance model that we will build in the next iteration. For implementing the language model, we experimented with two different types of models and evaluated their results first on a sample dataset of movie conversations.

6.1 Language Generation Model

In this section, we briefly explain the models that we built and evaluated for our language generation model.

6.1.1 Models

6.1.1.1 Sequence to Sequence using LSTM

A common sequence to sequence model is build up of two main parts, one is the encoder and the second is the decoder. Both the encoder and the decoder are neural networks that are connected with each other to form a sequence to sequence model. An encoder in case of this model takes an input as a sequence of words like for example, “Which mobile phone do you use?” Encoder encodes this information into a low level representation (low dimensional). This representation is then feed forward to the decoder which decodes this information into high level representation or in sentence like for example, “I use an iPhone 6”. We built a sequence to sequence models made up of LSTM cells.

We built a 2 layer model with each layer consisting of 256 LSTM cells. We trained it on the sample dataset of movie conversation with vocabulary size of 660 of most occurring words. Hot encoded format based on the vocabulary size was used for representing each word. Input sentences were hot encoded which were padded with [‘EOS’] (end of sentence) and [‘PAD’] like entities and fed into the network (encoder). After that output sentence was predicted also as a hot encoded vector. This hot encoded vector was converted back into a sequence of sentence. We trained on this sample dataset and recorded our results.

Here is a sample snippet of the comparison of the Real answer and the predicted answer by the model.

```
row 1
QUESTION: that is right
REAL ANSWER: i see
PREDICTED ANSWER: i is you

row 2
QUESTION: no there's no way
REAL ANSWER: hear me out
PREDICTED ANSWER: you me

row 3
QUESTION: oh yes
REAL ANSWER: when may i ask
PREDICTED ANSWER: i is you

row 4
QUESTION: do you believe it
REAL ANSWER: should i
PREDICTED ANSWER: i you
```

As it is self-explanatory that the results of this model were not very good.

There were some limitations of this model that we came to know. Some of the prominent ones are the following:

- Hot encoded format of dictionary was used for the word representation in this model. The main weakness of this format is that it treats each word as altogether into itself and it doesn't allow an algorithm to easily generalize cross words. This is because any product between any two one hot encoded vector is zero. That's why the language model was not quite able to associate relation between the words, which is the essence of the language.

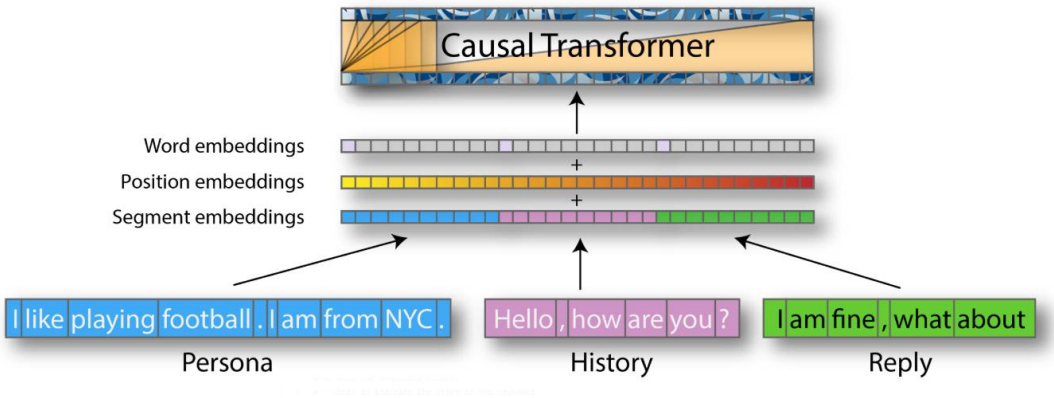
- Limitation of size of the data set was also there because used a small sample dataset. But even with the bigger dataset the results would have not been better because of the domination of previous limitation.

After, considering these limitations and consulting with our supervisor. We decided to move to the state of the art model named "Transformers". Transformers have proven themselves in the field of NLP by providing state of the art results in many NLP tasks like language modeling, machine translation, question answering etc.

6.1.1.2 Generative Pre-trained Transformer

GPT (generative pre-trained transformer) is a simple model that was trained on a bulk of web text for a simple task "next word prediction". This is a very powerful transformer and can generate long continuation of text by capturing the context of the text being produced. We are using language model that can produce predict the output word by taking input a sequence of words and generating a probability distribution over the vocabulary.

Pre –training is a very costly operation of such model on a hue dataset that is why we have fine-tuned the GPT model on our custom dataset. Since improving the consistency of the conversational agent is main objective of or project. Instead of just using a consistency assurance model with the language model, we are using the persona sentences of the agent also as the input context. There are mainly three input contexts, the persona sentences, conversation history and since our model generates tokens word by word we are also using the already generated tokens also as input context. Input sequence is made up of these context. We have used word embedding, position embedding and segment embedding for feeding all three contexts into the transformer.



We have used a next step prediction loss and language model loss and added them together. After fine-tuning the model on our dataset. The model was able to predict the next sentence tokens. For converting these tokens in to a complete response sentence, we have used a decoder mechanism. The most common method is the greedy method in this regard and beam search, which is a little more complicated and efficient.

In greedy method we choose the next token which has the most probability. While in the case of beam search a beam of sequences is maintained and in the end the best sequence is used as the output response. While these methods are good but there are some limitations to these methods as they fail to produce the distributional aspects off human text.

The results of top-k and top-p sampling are better than these old methods. Top k sampling keeps the token that fall in certain range of probability and discard other tokens. It help the model from going of topic.

.

References

[1] G. Z. Tomas Mikolov, "Context Dependent Recurrent Neural Network Language Model," in *IEEE*, 2012.

[2] P. M. Kory W. Mathewson, "Improvised Theatre Alongside Artificial Intelligences," in *AAAI*, 2017.

[3] D. M. T. C. H Nguyen, "A neural chatbot with personality," in *semanticscholar.org*, 2017.

[4] J. W. A. S. K. C. Sean Welleck, "Dialogue natural language inference," in *arxiv.org*, 2019.

[5] J. F. N. P. A. N. Tom Bocklisch, "Rasa: Open source language understanding and dialogue management," in *arXiv.org*, 2017.

[6] M.-W. C. K. L. K. T. Jacob Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv e-prints,* p. arXiv:1810.04805, 2018.