Confusion Modelling - An estimation by Semantic Embeddings

Praveen Mohanprasad

D18128998

A dissertation submitted in partial fulfilment of the requirements of
Technological University Dublin for the degree of
M.Sc. in Computer Science (Data Analytics)

2020

DECLARATION

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Analytics), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Technological University Dublin and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: ____M.Praveen Mohanprasad_____

Date: 01 September 2020

ABSTRACT

Approaching the task of coherence assessment of a conversation from its negative perspective 'Confusion' rather than coherence itself, has been attempted by very few research works. Influencing Embeddings to learn from similarity/dissimilarity measures such as distance, cosine similarity between two utterances will equip them with the semantics to differentiate a coherent and an incoherent conversation through the detection of negative entity, 'confusion'. This research attempts to measure coherence of conversation between a human and a conversational agent by means of such semantic embeddings trained from scratch by an architecture centralising the learning from the distance between the embeddings. State of the art performance of general BERT's embeddings and state of the art performance of ConveRT's conversation specific embeddings in addition to the GLOVE embeddings are also tested upon the laid architecture. Confusion, being a more sensible entity, real human labelling performance is set as the baseline to evaluate the models.

The base design resulted in not such a good performance against the human score but the pre-trained embeddings when plugged into the base architecture had performance boosts in a particular order from lowest to highest, through BERT, GLOVE and ConveRT. The intuition and the efficiency of the base conceptual design is proved of its success when the variant having the ConveRT embeddings plugged into the base design, outperformed the original ConveRT's state of art performance on generating similarity scores. Though a performance comparable to real human performance was not achieved by the models, there witnessed a considerable overlapping between the ConveRT variant and the human scores which is really a great positive inference to be enjoyed as achieving human performance is always the state of art in any research domain. Also, from the results, this research joins the group of works claiming BERT to be unsuitable for conversation specific modelling and embedding works.

Key words: *Semantic embeddings, Distance vector, Human score, Confusion, Coherence, Relevance, Encoder, Similarity*

ACKNOWLEDGEMENTS

I would like to render my sincere thanks to my supervisor, Dr Robert Ross for extending his complete support since the start of the dissertation. The ideas that he suggested, knowledge that he transferred, encouragements and the instant reviews that he rendered, are the reasons that I was able to complete the research successfully. I would also like to thank Dr Luca Longo for his guidance through the research principles.

I would also like to thank the Phd student of TU Dublin who acted as the external annotator for labelling the instances of the dataset.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

LIST OF ACRONYMS

NLP	Natural Language Processing
CBOW	Continuous Bag of Words
GLOVE	Global Vector
BERT	Bidirectional Encoder Representations from Transformers
ConveRT	Conversational Representations from Transformers
RNN	Recurrent Neural Network
CNN	Convolution Neural Network
CRISP-DM	Cross-Industry Process for Data Mining
LSTM	Long-Short Term Memory
GRU	Gated Recurrent Unit
NLTK	Natural Language Tool Kit

1. INTRODUCTION

1.1 Background

Communication plays a significant role in day-to-day world. Effective communication helps to comprehend the people in a better way. Communication has its own developing phase. The evolution of communication is from human-human interaction to human - machine interaction (Guzman, A. L., 2018, pp. 1-28). Machines, in form of computers and artificially designed agents are involved to interact with human. Many developing technologies are implemented for effective digital communication.

Dialogue system is a huge development in digital communication technology. Dialogue system is generally classified into two types: Task-oriented dialogue agent and conversational agent. Task oriented dialogue agent is designed to perform a certain task and to carry short conversations to extract some information in order to complete the task. Tasks like reserving a ticket for a movie or an event, finding products and providing personalized recommendations in online shopping are automated by these agents. Existing examples for a task oriented dialogue agent are digital assistants like Siri, Alexa, Google Now etc (Jurafsky, D., & Martin, J. H, 2018). Task oriented dialogue agents have widespread applications such as inventory control, tracking an order, building an email, framing a sales strategy and in marketing life cycles.

Conversational agent is a software application equipped with artificial intelligence which is developed and trained to simulate communication with human. Conversational agent acts as a great communicating partner by answering the questions. Conversational agents are designed to extend the conversation or chat, which is a characteristic feature of human-human interaction, rather than performing a particular task like booking a ticket for an event. Conversational agent is designed to process, return, and exchange requests (Pasupalak et al., 2017). Many conversational agents are developed in order to receive a request in form of text or audio to generate a response in an ordinary natural language.

The role of conversational agent in digital communication has its wide range of applications (Brandtzaeg, P. B., & Følstad, A, 2017). For example, many organisations deploy in conversational agents to provide high quality service and

guidance in customer support field. Customer support assistants who are actual humans can be replaced with intelligent and quick conversational agent. Conversational agents can respond instantly which can be of immense support in building the remarkable customer experience. Conversational agent can also be implemented in help desk, website navigation and technical assessing to diagnose any issues. It replaces the human effort at many instances where the humans are just used to perform a sequence of steps for a task or render template-based responses for a request.

These agents do a pretty good job when the response or a task falls into a pre-defined template. These templates represent the repository of responses stored from which suitable response is to be picked and rendered in case of retrieval based chatbots (Ramesh, K., Ravishankaran, S., Joshi, A., & Chandrasekaran, K., 2017). In case of generative chatbots, these templates represent the pattern and scope of trained request-response pairs and the chatbot renders a response based on this training. Challenges occur when the requests fall apart from the defined templates where the chatbot renders out irrelevant responses (Shum, H. Y., He, X. D., & Li, D., 2018). There is a need to redefine design parameters of an agent to overcome such challenges (Clark et al., 2019).

A cold experience occurs when interacting with conversational agent which lacks to meet some expectations. Interaction with an agent is robotic and also clunky which makes the user frustrating. Some conversational agents are not so effective in providing space for any deviations occurring outside the trained script. Every individual has their unique way of natural language like command over language style, slang of words, habit of misspelling certain words, short forms etc (Rahman, A. M., Al Mamun, A., & Islam, A., 2017). At times, conversational agent finds it difficult to interpret these kinds of basic comments or requests. A Conversational agent will not able to understand these kinds of comments, gives inappropriate responses or answers to the user. User gets confused when he or she receives an irrelevant response. Hence conversation becomes unnatural and humans find it difficult to relate to one’s own request. This type of communication gap where the request and the response semantically stay in different directions not only happens in human-agent conversation but also in a human-human conversation where one human renders an irrelevant

response misunderstanding the request or statement of another human. Irrespective of the partners involved, the context in terms of confusion and irrelevance stay the same.

1.2 Research Problem

An irrelevant response produced by some low trained conversational agent leads to confusion. Human gets easily confused when the response is totally irrelevant to the request or query and feels a very unpleasant experience. The occurrence of confusion interrupts the flow of conversation and there is also possibility for the termination of conversation resulting in user's dissatisfaction. As shown in Figure 1.1, the user was narrating about the experience of visiting London, while the conversational agent gave a response about eating food which is completely irrelevant to the user's statement. This makes the user to get confused and feel disturbed on the conversation flow. The coherence of the statement gets disturbed when an irrelevant response is produced which leads to confusion of the human. The coherence of a conversation mainly depends on the relevance between the request and the response. The three terms coherence, relevance and confusion are tightly bound to each other. To put in a particular order, it can be stated that, higher the relevance between a request and a response, Higher will be the coherence of the overall conversation and lower will be the confusion for the participants involved in the conversation.

Figure 1.1 Conversation between a human and conversational agent causing confusion in the human participant

The coherence of conversation can be measured by means of presence or absence of confusion. To achieve a coherent conversation, where there is a good relevance witnessed between the request and the response, the confusion causing response should be avoided by the partner. To achieve this, first the confusion in the conversation should be spotted. Machine learning can be implemented to develop a model or system which is designed and trained to spot the confusions in conversation. When the intelligence is able to spot the confusion successfully, two positives emerge, one is confusion causing responses can be avoided beforehand by a human or agent and the other is responses can be ranked by the degree of confusion presence and the response with the lowest rank of confusion presence or highest rank of similarity can be delivered by an agent (Henderson et al., 2019).

The distinct intention of such model is to classify the conversational instances of human conversation with conversational agent as confusion or 'no confusion'. Embeddings is a concept in NLP, where every word in a sentence is represented by a vector of n dimensions. In a well-trained embedding space, the relevant words will be close enough while the irrelevant words with contextual differences will stay apart (Kiela, D., Hill, F., & Clark, S., 2015).

The supposition of this research is that distance between such semantic embeddings can be used to spot confusion in a conversation by classifying such instances as 'confusion'. This process of detecting confusion in conversation between any two specific partners can be used to decide the coherence of a conversation in general irrespective of the partners type, human-human or human-agent. This way of detecting confusion by means of semantic embeddings will be able to achieve a human comparable performance as how the humans are naturally able to spot the irrelevance or confusion in a conversation.

Based on the discussed challenge and the possible solution, the research question becomes,

Is it possible to measure the coherence of human conversation with a conversation agent by means of semantic embeddings and achieve a human comparable performance?

1.3 Research Objectives

The research objective of this research can be framed from the hypothesis. So, the hypothesis is first defined. The positive side of the research question of this research states that the confusion in a conversation can be detected by means of semantic embeddings and the obtained performance in confusion detection process will be comparable to the real human's ability of spotting out confusion. To accept the hypothesis, the proposed model's performance on detecting confusion should be approximately equal or nearly equal to human's performance. Experiments are done to test the defined hypothesis and evaluated using certain metrics. These metrics are nothing but the general evaluation metrics of any machine learning model such as accuracy, recall, precision and f1 score. These metrics will be used to statistically measure the so called 'comparableness' which in other words, 'equalness' or 'near/approximate equalness' of the proposed method with human performance. 'Recall' is chosen as the metric of comparison. Reason for the choice is discussed in the section 'Evaluation' in the next chapter. Hence the alternate and null hypothesis become,

- **H1:** If the coherence of human conversation with a conversational agent is measured by means of semantic embeddings, then the recall of the proposed model is statistically equal to that of human performance
- **H0:** If the coherence of human conversation with a conversational agent is measured by means of semantic embeddings, then the recall of the proposed model is not statistically equal to that of human performance

Figure 1.2 denotes the graphical representation of the above defined alternate hypothesis (H1).

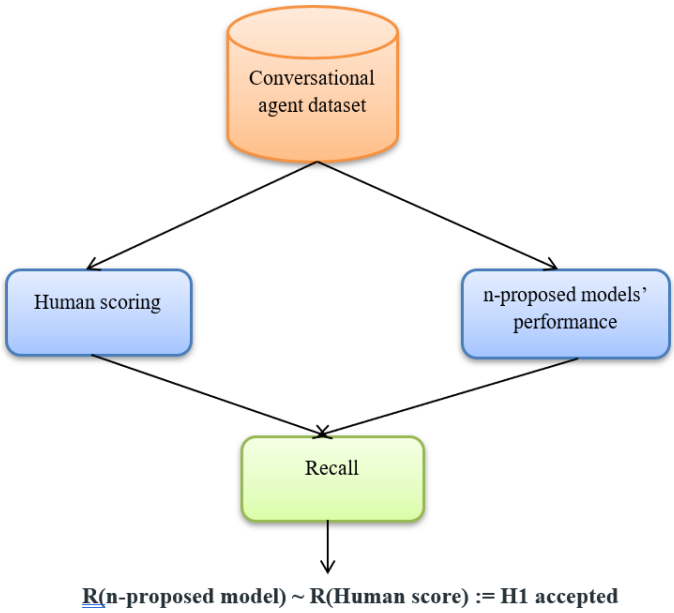

Figure 1.2 Graphical representation of research hypothesis

The research objectives are now clear from the hypothesis defined,

- To investigate the current and related works on confusion and coherence, state-of-art works on embeddings with respect to the domain
- To identify a conversation type dataset and do all the required pre-processing steps, shape it and make it compatible for confusion modelling.
- To define the baseline performance which are human scores
- To design and build the proposed model architecture that learns the semantic embeddings.
- To conduct the experiment on all variants of the proposed architecture, display the results on recall and find the best performing variant.
- To evaluate and compare the best forming variant and the baseline human performance on their recall scores.
- To statistically test the comparison and accept/reject the alternate hypothesis.

1.4 Research Methodologies

This research can be categorised as empirical, secondary and quantitative research. First, a detailed literature review with respect to the research domain is undertaken. With the knowledge gathered out of the review, a research question is framed in form of a theory. Hence it takes the form of an empirical research. The theory is that Confusion in a human conversation with a conversational agent can be spotted by using semantic embeddings and the gained performance will be comparable to human performance. As seen, the theory represents a supposition which is defined as the hypothesis. Experiments are conducted to test the hypothesis through some numerical metrics. Hence the research is categorised as quantitative type. The dataset used in the experiment was generated at the Conversational Intelligence Challenge which took place in July 2017, hence it takes the form of a secondary research. It falls into deductive reasoning category as the overall direction is Theory – Hypothesis – Observation – Confirmation. First it starts with a theory as stated before, a supposition is taken which is framed as hypothesis, experiments are done to test the hypothesis and observations of measurable variables are made and finally the noted observations are tested statistically to accept or reject hypothesis. This is purely a deductive reasoning direction.

1.5 Scope and Limitations

Confusion detection and coherence measurement of human conversation with a conversational agent has two segments of research work. One is detecting confusion in the human conversation with a chatbot and the other is including such detection process, to build a chatbot which would predict confusion of the human participant and generates a relevant response to make the overall conversation a coherent one. The undertaken research only deals with the first part. The scope is clearly to only detect confusion in human conversation. Coherence and confusion is a wide area or research that has applications over essay scoring, text summarising etc. coherence can be measured even on a paragraph, bunch of text sentences, a document etc to check how relevant the sentences are to each other in a document or an essay. But this research deals only with conversation type text, to measure the coherence as how good/relevant a response is with respect to a request. The scope is only over the conversational domain.

Across several techniques, The NLP technique used here is only embedding and its semantics. CNN have also been proved to work well with text sequence-based research works other than RNN. This research only deals with RNN type networks for all the variants involved. There are a lot of pre-trained embeddings in NLP space but the research experiments only a few pre-trained embedding type such as BERT, GLOVE and ConveRT. Though the research is experimented with human-agent conversation data, it can extend its application and scope over human-human conversation as well.

Limitations of this research work starts with the size of the dataset first. The ConvAI.io dataset has a total of 4750 dialogues. But only 800 instances are used for training the models. Also, the dataset has both single turn and multi turn conversations, but only single turn conversations are picked from the entire dataset. Lengthy sentences are ignored while picking up the samples. Next limitations come from inadequate computational resource. This restricts the experiments in the process of fine tuning huge embedding space of BERT further after initializing.

1.6 Document Outline

The remainder of this document is structured through the sequence of following chapters,

1.6.1 Chapter 2 - Literature Review

It records the reviewed literature with respect to the research problem. A wide literature review on evolution of communication between human and machine, Techniques in NLP to process natural language, Topics on embeddings, its types and all the available pre-trained embeddings, works on the relationship between the similarity metrics and embeddings, Different architectures of neural network designed to train semantic embeddings. This is followed by reviews on the literature which dealt with this research problem of coherence modelling and confusion detection. A comparison is laid out between these works and the gaps are listed.

1.6.2 Chapter 3 - Design and Methodology

It includes the adopted CRISP-DM research methodology and details on how this research is driven through its various stages in detail such as Data understanding, Data preparation which includes all the pre-processing techniques implemented on the

identified dataset and their results, Modelling which includes the components of the proposed base design and the design of all the base model variants to be experimented and its final stage of evaluation to be done using the right evaluation metric.

1.6.3 Chapter 4 - Results, Evaluation and Discussion

It includes the information on the execution of experiments on the baseline performance and all the other variants proposed in the previous chapter. This information includes the nuances of training of different models, challenges faced during the model training, adopted hyper parameters during the training, Results of the evaluation of different variants and their comparison against the base line performance with statistical tests to accept/reject hypothesis and a brief discussion on the results.

1.6.4 Chapter 5 - Conclusion

It summarises the overall research problem dealt, the design and the experiments done and comments on the final results obtained. It mainly focuses on the contribution and the impact that the research has left in the domain of study on a narrow perspective and narrations on all possible leverages on future works, recommendations for the other researchers from the experience obtained.

2. LITERATURE REVIEW AND RELATED WORK

2.1 Background

The recorded literature goes through the research works in a sequence related to evolution of human machine communication, Techniques involved in NLP to improve the process natural human language, Types and role of embeddings in NLP, Usefulness of pre-trained embeddings, Relationship between similarity metrics and embeddings, Neural network architectures to learn efficient embeddings, related works to confusion detection, coherence modelling and the gaps are identified.

2.1.1 Evolution of Human - Machine Communication

It is not so complex for a common man to understand the natural language but at the other end, dialogue system or a conversational agent finds it difficult to interpret and process this kind of human language at times of human–machine communication. So, human started to change some of his or her natural language style while interacting with machine (Hill, J., Ford, W. R., & Farreras, I. G., 2015). This mainly lacks the richness in vocabulary of human-machine interaction. Hence, technology started focusing on learning about how to process the natural language and thus the natural language processing has been developed. Natural language processing plays a major role in the evolution of human - machine conversation (Bird, S., Klein, E., & Loper, E., 2009). The ability of a conversational agent to effectively understand and process the natural language has a great impact on extending the usage of conversational agent in communication field.

A computer program is designed and developed to interact in a conversation with human in natural language and it is known as conversational agent or chatbot. It is clearly defined as the conversational agent is an artificial intelligent model that is used to communicate with human in their own language (Niculescu et al., 2014). The medium of language can be of any patterns like audio, text etc. In the year of 1960, ELIZA is one of the conversational agent that are developed is trained to receive the data as input and scan the input data for the keywords (Weizenbaum, J., 1966, pp. 36-45). There are main three components that are included in the construction of the conversational agent. They are: NLU (natural language processing), dialogue

management (DM), and natural language generation (NLG) as seen in below Figure 2.1 (Wang, X., & Yuan, C. , 2016).

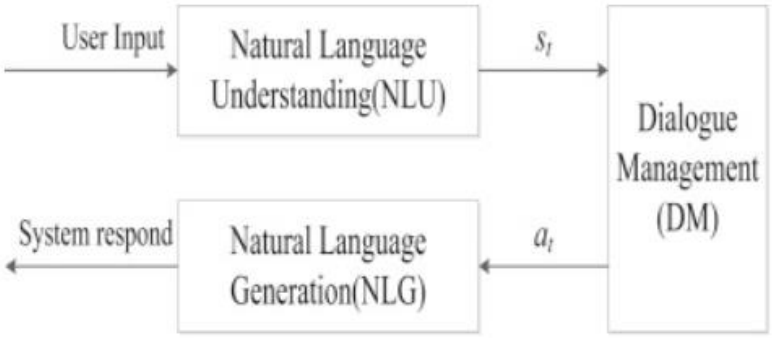

Figure 2.1 A basic frame for dialogue system when states are fully observable (Wang, X., & Yuan, C. , 2016)

2.1.2 Techniques Involved In NLP

There are various techniques involved in processing the natural language. The initial step in natural language processing is tokenization. In simpler terms, tokens are defined as the smallest individual units. Any alphabet, number or even punctuation marks can be treated as tokens and are processed (Webster, J. J., & Kit, C., 1992, pp. 1106-1110). Stemming and Lemmatisation are the two different process enrolled in normalisation of lexicons (Gharatkar, S., Ingle, A., Naik, T., & Save, A., 2017, pp. 1-4). Stemming is the process of identifying the related words and mapping into its root word. The target word is processed to remove the matching and trailing characters in algorithm dataset.

The lemmatisation identifies the diversity in acronyms of a root word and maps them to its root word family. The lemmatisation algorithms are embedded with logical analysis of words using dictionary (Zeroual, I., & Lakhouaja, A., 2017, pp. 1-6). Noise reduction is similar to lexicon normalisation because both the techniques determine to reduce the inputs in natural language which is very helpful in improving the accuracy of machine learning models (Furlan, B., Batanović, V., & Nikolić, B. , 2013). Another effective method of reducing noise in NLP is identifying the similar entities and removing or replacing them.

2.1.3 Role of Embeddings & Pre-Trained Embeddings

Machine learning models find the vector representation as a simpler and effective form of data and process them efficiently. In bag of approach, every word is considered separately and hence it is difficult to identify and process any pattern. As a solution to this, another method called n-gram is used that groups the n number words in a sentence into a single dataset. Bigram deals with sequence of two words where trigram is the sequence of three words (Alberto, T. C., Lochter, J. V., & Almeida, T. A, 2015). Word embedding is a technique where pre-processed text is represented in vector form. These vector representations are learned in un-supervised manner by model that is designed using the concept of machine learning. For example, in Word2Vec, every text is considered as an input and it generates an equivalent vector space which is considered as output. In word embedding, each unique text is represented using a vector dimension.

In vector representation, the location proximity of the associated words is close. Word2Vec uses two types of model architecture to generate an output namely: continuous bag of words (CBOW) and continuous skip gram (Mikolov, T., Chen, K., Corrado, G., & Dean, J. , 2013). The results of word2vec are better compared to that of in-hand optimised fine-tuned vectors (Kim, Y., 2014). Multiple degrees of similarity have been identified for words which lead to represent a phrase, having a bunch of words, by a single vector (Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J, 2013).

Skip gram method introduced is a supervised technique to learn word embeddings from a text corpus where, for an input word, target words are chosen by skipping few words ahead or behind the sentence and trained to form n-dimensional vectors for words while predicting those chosen target words whereas, CBOW is an opposite technique to this where one word is chosen as a target word and embeddings are learnt by predicting that one word by using many surrounding words as inputs as seen in the Figure 2.2 (Mikolov et al., 2013). The problem with this type is that the global occurrence of the words is not taken into account and formation of input and target words are restricted to a sentence. Glove stands for the term ‘Global Vector’ overcame this limitation by considering the global co-occurrence of similar context words (Pennington, J., Socher, R., & Manning, C. D., 2014). There are more than hundreds of

dimensions for a word or text when it is converted in vector representation. The main objective to develop glove model is to combine the principles of both cbow and continuous skip gram model which results in providing an effective algorithm which is more superior and technical (Sharma, Y., Agrawal, G., Jain, P., & Kumar, T. , 2017).

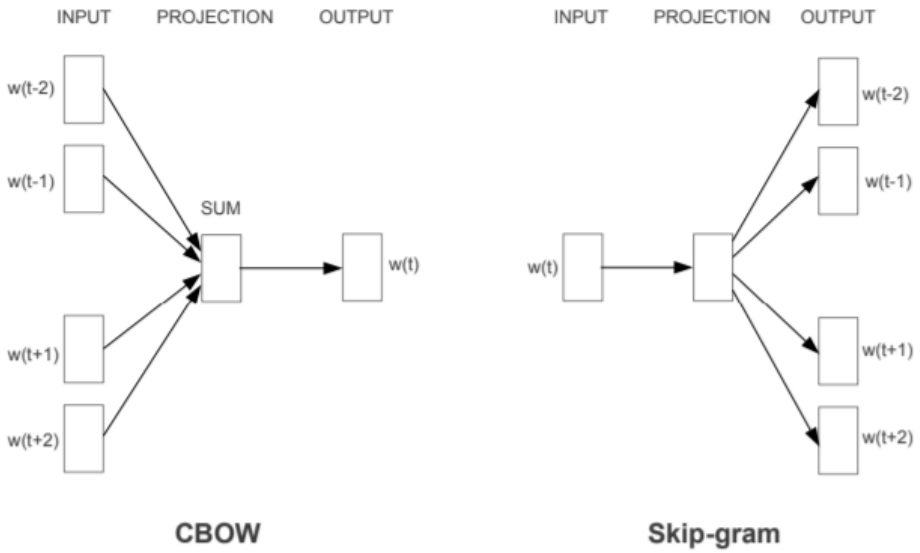

Figure 2.2 Many to One CBOW & One to Many Skip-gram techniques

(Mikolov et al., 2013).

Obj2vec is an extended form of word2vec. Word2vec is vector representations of words where as obj2vec is vector representation of not only words but any paired objects like a sentence-sentence, customer-product, movie-genre etc. Inspired by skip-gram model, It is an embedding space trained by symmetrical objects (Islam, M. M., Sarkhel, S., & Venugopal, D., 2019).

2.1.4 Similarity Metrics & Embeddings

Glove word embedding vectors are based on the concept of using Euclidean distance or cosine similarity to find the nearest neighbours for any word vector and there are three variants of it as 50, 100 and 300-dimensional glove embedding vectors (Pennington et al., 2014). Similar concepts of cosine similarity is used in ConveRT (Henderson et al., 2019) and IRIS (Banchs, R. E., & Li, H., 2012) chat oriented systems to select response for the given user input from the database where every response is represented by a vector, the response is thrown out by selecting the one having high similarity score with the input vector. As seen in the Figure 2.3, the final

embeddings are compared at the utterance level and not word level on the cosine similarity scores and then ranking of responses based on the similarity score is done to select the most relevant response from the repository.

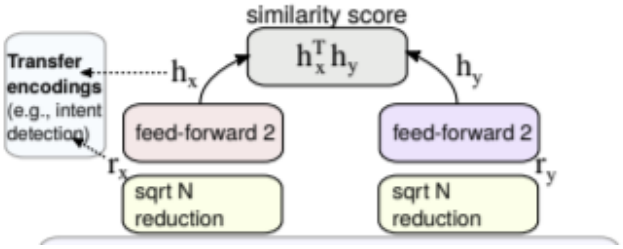

Figure 2.3 Cosine similarity in the final layer to learn embeddings

(Henderson et al., 2019).

Even in the world of machine translation the concept of Euclidean distance is used. In a many to one translation system, clustering of input languages happens based on the Euclidean distance to allow shared parameter architecture. The respective translation words of multiple languages form a cluster as the Euclidean distance between those vectors in the embedding space is low (Tan et al., 2019). Similarly, it has shown good improvement in the performance of text classification by reducing the sparsity caused by short sentences just by replacing the short text representations with the one closer to them. This closeness is decided by the Euclidean distance (Wang, X., & Yuan, C. , 2016).

Cosine similarity is used as a parameter to control the response, making it relevant. The cosine similarity attribute feeds the bot with similarity score between the request and the predicted response. The bot learns to provide responses in a motive to maximise the similarity score (See, A., Roller, S., Kiela, D., & Weston, J, 2019). The cosine similarity attribute feeds the bot with similarity score between the request and the predicted response. It is a better approach that when the similarity score is added as another parameter, the bot learns to provide responses in a motive to maximise the similarity score. Thus, the similarity metrics of Euclidean distance and cosine similarity have been widely used with respect to semantic embeddings.

2.1.5 Neural Architectures to Train Embeddings

An effective deep learning algorithm is recurrent neural network (RNN) in where, data or text move forwards only in one direction. This data flow acts as a feedback for another layer (Lai, S., Xu, L., Liu, K., & Zhao, J., 2015). The inputs are memorised and then used for prediction. This network is mainly used in the field of speech recognition, translation etc., where there is a solid output as prediction. Also, they are used in training word embeddings. The LSTM (Long Short-Term Memory) differs from the basic RNN concept by holding or storing the data for very long time.

The LSTM can add or remove the information to the layers using structures called as gates. The LSTM network is specially designed and trained to handle long range dependencies that occur in lengthy sentences especially in a conversation type text, where a single conversation has multiple utterances/sentences. GRU (Gated recurrent unit) is very similar to the LSTM mechanism. LSTM and GRU both have a 'forget gate' but output gate is present only in LSTM. This also made LSTM to have more parameters to be learnt than GRU. In the comparison between GRU and LSTM it is stated that the LSTM mechanism is more effective because it has the capability of handling the unbounded counting (Pitsilis, G. K., Ramampiaro, H., & Langseth, H., 2018). Still some researches prove that GRU performed better (Fu, R., Zhang, Z., & Li, L., 2016).

Seq2Seq type recurrent neural networks have an encoder-decoder based architecture which can be used to generate sequences and not just words. Encoder takes the input sequence as one word at every step and decoder generates the output sequence as one word at every step as seen in Figure 2.4. These type of networks can be used in applications that include machine translation, caption generation etc. They work on the concept of maximising the conditional probability of the target sequence given the input sequence (Sutskever, I., Vinyals, O., & Le, Q. V., 2014). These Seq2Seq networks struggled in the process of encoding long sentences into fixed length vectors without losing any information. Seq2Seq with attention type networks allowed decoder to produce output at every timestep by means of different weightage allotted to encoder outputs at respective timesteps. This had a great improvement on Seq2Seq networks with respect to producing more relevant outputs without losing information in long sentences (Cho et al., 2014).

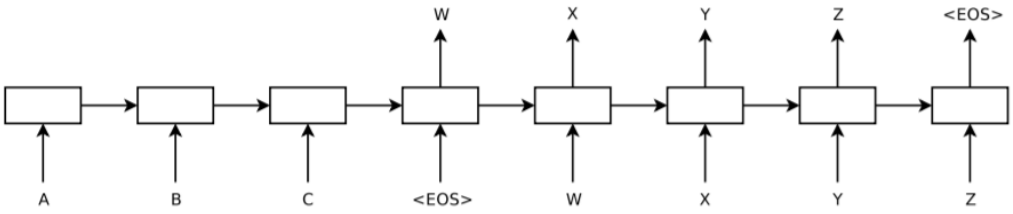

Figure 2.4 A Encoder - Decoder architecture of seq2seq network

(Sutskever et al., 2014)

A great impact on learning embeddings was left by the development of transformer attention-based model embedded with multi head and self-attention mechanism (Vaswani et al., 2017; Casanueva, I., Temčinas, T., Gerz, D., Henderson, M., & Vulić, I., 2020). In general, most of the seq2seq attention-based models are capable of paying attention to the input sequence from the encoder at every position. But this transformer is designed in a way to apply the attention and importance to not only the encoder position, but also to the previous decoder layers at every position. BERT (Bidirectional Encoder Representations from Transformers) is based on a encoder only attention based transformer network, parses the input sequences on both the directions, generates word representations that overcome the limitations of word2vec and glove which allots same fixed vector to a word irrespective of the context, it is used. BERT accepts a pair of sentences separated by a token and some of the words in these sentences are masked, the network learns embeddings in the way of predicting those masked words (Devlin, Jacob, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, 2018) as seen in Figure 2.5.

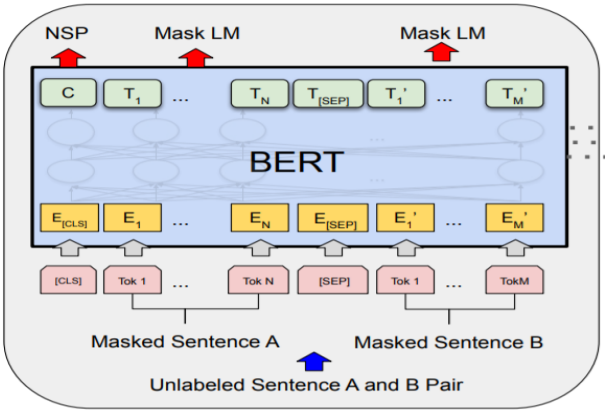

Figure 2.5 Transformer, encoder only network of BERT

(Devlin et al., 2018)

ConveRT (conversational Representations from transformers) is the state of the art performant in the response selection task, has a dual encoder attention based transformer architecture with a concatenation layer outperforms BERT's embeddings in the conversation domain with respect to training cost and time and is also light weight. Differences in conveRT's as compared to BERT will be trainable positional matrices instead of fixed positional embeddings, absence of the process of masking tokens, presence of a concatenation layer and BERT like general purpose embeddings is not suitable for conversation domain (Henderson et al., 2019). Another dual encoder architecture was used to train sentence embeddings. The performance of this universal sentence encoder (Cer et al., 2018) is close to the performance of ConveRT.

2.2 Related Work

2.2.1 Confusion detection - A user feedback dependant

Reasonable works have been contributed with respect to confusion detection in conversation by chatbots, coherence assessment & modelling in text sequences. In a human conversation with an intelligent agent, when the agent throes out irrelevant responses, the human participant gets confused and responds back with some template statements or questions. These templates would be something like 'what are you talking about?', 'what do you mean?', 'I think you did not get me'. In case of relevant responses given by the agent, the human participant would leave statements like 'oh great!', 'Thanks for the assistance' etc. These human responses are used to detect the confusion in the conversation which is called as absurdity (Hashimoto, C., & Sassano, M., 2018). The researchers grabbed a mix of these conversations from the log and built a conversation classifier to classify if the conversation is absurd or not.

There can be seen a clear challenge here which is the dependency on the human's response in a similar fashion as expected in a good or absurd conversation. As seen in the Figure 2.5, the scored feedback utterance DB stores all the feedbacks of the user that are already scored as bad or good feedbacks and the system looks up against this sample storage whenever it gets the new dialogues from the log. There are high chances that the human response would be out of the scope of defined or learnt templates stored in the scored feedback utterance DB which is so far seen by the

classifier in the log. Hence, there is a high risk of ending up in low recall performance of the developed classifier.

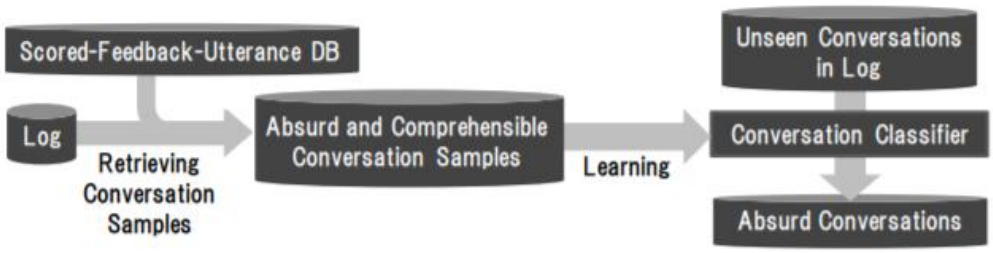

Figure 2.6 Absurdity detection using user's feedback utterances

(Hashimoto, C., & Sassano, M., 2018)

Another research was done to detect student's confusion in online courses by means of rule-based hashtags. The learnings obtained from the hashtags of the confused courses are used to predict the confusion in the courses that do not have hashtags again; this is purely dependent on the scope of hashtags that the model learns at the start (Geller et al., 2020).

The research work that measured the quality of the conversation from the negative side, by detecting the errors causing a possible breakdown in the conversation, did not end up in a success (Higashinaka et al., 2015). The human to bot chats were collected and were labelled by 124 human annotators as 'no breakdown - 0', 'possible breakdown - 1', 'breakdown - 2' and then based on the comments (reason of breakdown) given by the annotators the conversational errors were clustered and put into categories and then correlation was computed between the categories and the degree labels. Higher the correlation, higher the degree of breakdown. The downfall of this work as compared to the previous one is that it involves human annotators whose inter agreement cohen-kappa value turned out to be 0.26 which is poor. Also, conceptually, clustering the conversation based upon user comments about the reason for breakdown looks inefficient because the annotators were not limited to template based comments but left free to comment in their own way on a broader space which resulted in too many clusters having imbalanced number of comments.

Another research approached the same confusion detection but in an indirect method. User satisfaction was used as a parameter to measure the conversation and response quality (Kiselev et al., 2016). The researcher works out to predict the user satisfaction

by means of interaction signals. These interaction signals included users' click, touch and voice gestures. Users actions are monitored to decide if they are satisfied or not. Higher the satisfaction, higher the coherence and lower the confusion. Compared to the previous two research works, this looks better as the researcher uses user satisfaction as a parameter and tries to predict it using external elements such as interaction signals which is a bit improved way compared to limited scope of user's feedback utterance and external annotator.

Error detection in the human conversation with chatbot by means of positive and negative cues given by the human participant was attempted in another research (Krahmer, E., Swerts, M., Theune, M., & Weegels, M., 2001). These cues are given by the human in turn to the explicit and implicit questions asked by the agent. Negative cues denote that the response given by the agent is irrelevant and the human participant is confused. Negative side of this work is again user dependency on the error spotting process and the positive edge that this research has over the previous experiments is that this model automatically spots the errors in the conversation online using these cues.

The proposal for the usage of reward signals from the human to measure the quality of conversation had benefits over using cues (Meena, R., Lopes, J., Skantze, G., & Gustafson, J., 2015). This is more similar to the work of Hashimoto, C., & Sassano, M., (2018**).** Here the positive rewards and the negative rewards are quantified by the feedback of the humans to decide on the conversation quality. The second part of the research also works on learning from these rewards to improve the quality. The research that predicted the quality of speech recognition based on customer satisfaction within a session added enhancements to the work of Kiseleva et al., (2016). Quality of conversation in terms of confusion is measured by the action sequences within that particular session of the conversation, rather than by using interaction signals (Jiang et al., 2015). The added feature that this work contributed is measuring the quality of conversation over a session having a group of utterances and not over individual utterances.

Another research had an online active approach of detecting confusion on the fly, where self-learning bot seeks the user's response to label the confusion and re-train the model (Hancock, B., Bordes, A., Mazare, P. E., & Weston, J, 2019). All the above

works in spotting the confusion or measuring the quality of human conversation had one thing in common which is user dependency in the detection process.

2.2.2 Coherence Modelling – An entity dependant

Early works with respect to coherence involves various type of relations, entity based (Grosz, B. J., Joshi, A. K., & Weinstein, S., 1995; Barzilay, R., & Lapata, M., 2008) and lexical relationship based (Klebanov, B. B., & Flor, M., 2013; Somasundaran, S., Burstein, J., & Chodorow, M., 2014; Dong, F., Zhang, Y., & Yang, J. , 2017). Out of the entity based works, the work by Barzilay et al., (2008) is a popular one which converts a text sequence into a set of core entities first and based on the continuity of semantics of the spotted entities in subsequent sequences, the coherence is decided. Amount of changes in the distribution and syntactics of these entities in the fourth-coming sentences is inversely proportional to the coherence of the document until the compared sequence. Here, the entities are identified by doing a straight match over the core noun words of all the noun phrases in a sentence.

The entity grid model was later taken to its next level by some of the successive works (Feng, V. W., & Hirst, G. , 2012), by extending the phase of entity extraction of the previous work in a effort to improve the coherence modelling. They used more entity specific features to extend the grid. Entity specific features like named entity and noun classes are used to distinguish the entity types in the entity grid which contains not only the core head nouns but also the non-head nouns (Elsner, M., & Charniak, E., 2011). The objective of these features is to differentiate the core entities and the other unimportant entities in the sentence. These features are derived with the help of external corpus through the additional information from syntax and statistics of the similar entities between the external corpus and the actual sentence.

The original entity grid coherence model was also extended by means of syntactic patterns and discourse relationships which gained additional performance in coherence modelling through entity detection (Louis, A., & Nenkova, A., 2012). A different attempt on coherence modelling using convolutional neural network was done. The usage of CNN over RNN-LSTM and attention gained good performance in capturing long range entity transitions and aligned features in a sentence (Nguyen, D. T., & Joty, S. , 2017). Here, a pairwise ranking method is demonstrated where the model accepts two inputs and ranks them based on the coherence or relationship between them This

work also overcomes the issue of having the feature learning process completely detached from the downstream tasks of ranking or classifying which was seen in the previous discussed works. As it is recorded clearly on the challenges of using entity detection to decide on the coherence of text and how different researches were carried out to mitigate the same. These works had that extra dependent layer of entity identification.

The below Figure 2.7 clearly show the extra entity identification layer at the start in an entity grid after which the actual convolution starts

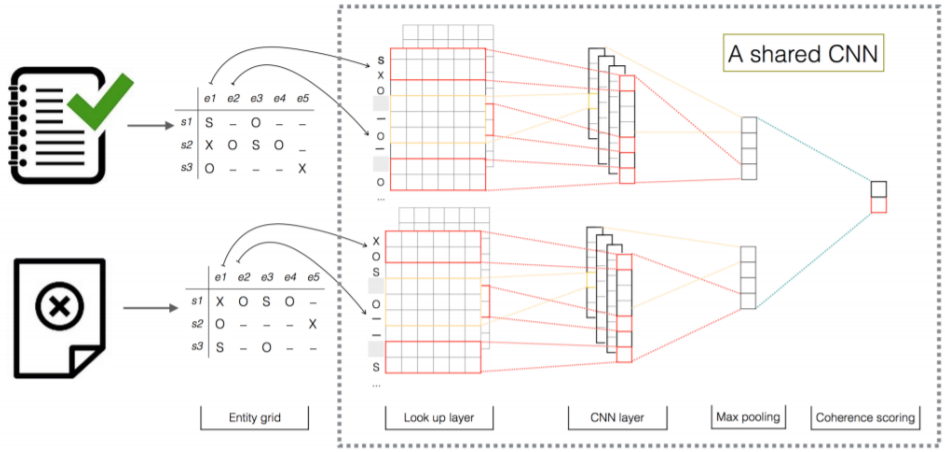

Figure 2.7 A CNN for coherence modelling by pre-processing entity grid

(Nguyen, D. T., & Joty, S. , 2017)

2.2.3 Coherence Modelling - A lexical relationship dependant

Similarly, another object called lexicons are used to measure the coherence of the text or essays (Somasundaran, S., Burstein, J., & Chodorow, M, 2014). This work used the concept of lexical chaining that connects the words that are related or relevant to each other. Two lexicon feature sets were constructed in prior to the coherence classification. Another concept of word association, also called as co-occurrence of words was used to score the quality, coherent essays (Beata et al., 2013). Essays containing high number of associated pairs and dis-associated pair of words are considered to be high scoring essays. Word association profile was first created using a huge training corpus and then the coherence of the texts was evaluated based on the percentage of associated words present in them. The tightness of the text on the topic was measured as coherence.

This research and the previous one are closely related to each other but the one that uses lexical chaining has added advantage of chaining more than 2 words and forming lengthier chains of related words which can efficiently break the limitation of coherence of a text getting incorrectly classified just by a having a single associated or multiple associated words instead of having a chain of associated words in no particular order. Another work based on lexical relationship was done where these relationships between two sentences are encoded by means of a graph, sentences are represented by nodes, semantic lexical relationships between the sentences are denoted by edges and coherence decision is deduced by subgraph filtering method that mines out the semantic relationships alone (Mesgar, M., & Strube, M., 2016).

A homogeneous work was carried out on the same concept of lexical relationship but CNN (Convolution Neural Network) was used instead of graphs and RNN layers wrapped words over to capture the overall context of the sentence whereas the graph based work considered words individually with no terms of the overall sentence context (Mesgar, M., & Strube, M. , 2018).

2.2.4 Coherence Modelling using embeddings

Some works on coherence modelling were with respect to embeddings. Sentence embeddings derived from RNN and a window of sentence vectors are accumulated and coherence of sentences within the window is decided by the probability. This coherence is evaluated by identifying the changes (too high or too low) in the aggregation value of the subsequent coherence probabilities obtained by sliding the window for the next set of sentences with a certain overlap with the previous window (Li, J., & Hovy, E., 2014).

A state of the art performance was achieved by a seq2seq type generative model in generating coherent sentences by means of latent dependencies captured by topic modelling to make sure that the generated sentence stays close to the topic or context of the that the input sentence. Again, sentence embeddings were used here as that of the previous work but latent vector representations are used here to decide the coherence while sigmoid probabilities were used to decide the coherence in the previous work (Li, J., & Jurafsky, D., 2017).

Research work on automatic essay scoring by using CNN and RNN to measure the coherence of text, makes use of character and word level embeddings to form a sentence vector in the text and decided the coherence again using a sigmoid layer as in Figure 2.8. It was found that using character level embeddings performed well over the word level embedding (Dong, F., Zhang, Y., & Yang, J. , 2017). This approach eliminates the usage of any elements like entities or lexicons but just the embeddings to measure the coherence of text.

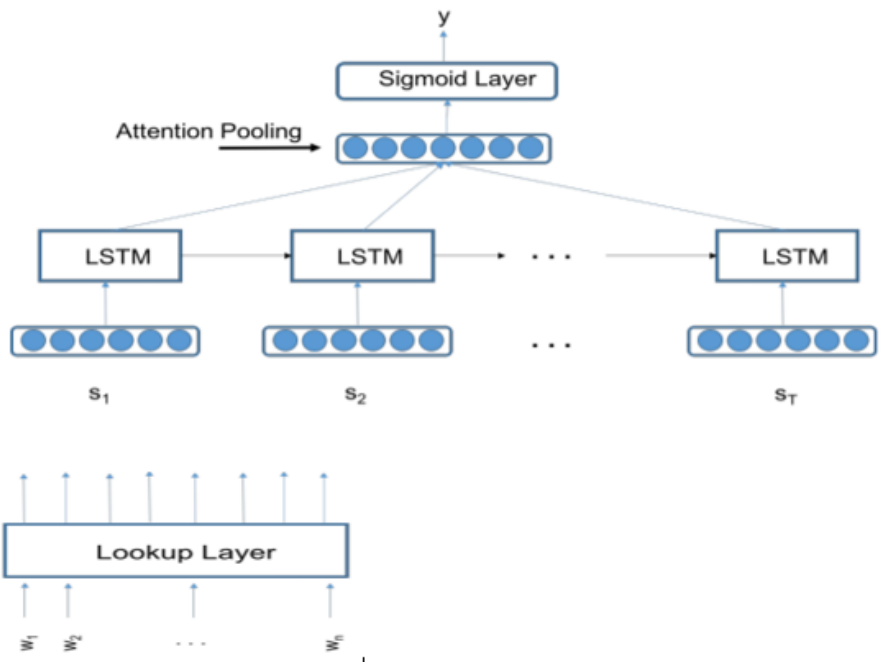

Figure 2.8 Formation of sentence embeddings through words to decide coherence (Dong et al., 2017)

2.3 Gaps in Research

Some of the current approaches to detect the human confusion in a conversation deal with strategies like detecting confusion with the help of participant's response (Hashimoto, C., & Sassano, M., 2018), detecting confusion with the help of user defined hashtags or keywords (Geller et al., 2020). All these restrict the system's ability to detect confusion to well-defined boundaries and templates. This reduces the generalisation capacity of the system by getting fit into some limited scope or template.

The other common problem seen in the other works of confusion detection is the system using user and external elements as intermediate assistance objects to detect the

confusion. Measuring the satisfaction rate of the user to detect the presence of confusion (Kiseleva et al., 2016). Next to user, the systems used other external elements as intermediates to detect the confusion. These external elements include interaction signals like click, touch and voice gestures .Using these elements enhance the scope from the previous template-based limitations, asses the presence of confusion quite better than the user feedback and the keywords but these elements cannot be easily obtained.

Obtaining such external knowledge to predict the success or failure (confusion) of the response makes the confusion detection processes an expensive one. In addition to interaction signals, using elements such as positive/negative cues (Krahmer et al., 2001) and rewards (Meena et al., 2015) is a tedious process where these elements are obtained from separate processes such as answering and analysing bots' feedback-based questions and processing user feedbacks respectively. All these approaches have one thing in common which is involving external knowledge to spot the confusion in a conversation. As stated earlier, the cost of obtaining this external knowledge is high.

When the cost of obtaining external elements becomes bearable to the research then it is better to use external annotators instead of the above mentioned objects. Though, external annotators are equally costly, the performance and accuracy of human annotators is always the state-of-the-art performance in any research. Also, getting direct confusion labels from human annotators need no further processing, just analysing the consistency is enough. Still, this method turned out to be a limitation in one of the works that used inconsistently labelling human annotators (Higashinaka et al., 2015). The same work over used human effort by using such annotators not only to score the confusion but also comment on the reason for confusion which demanded further processing of those comments which could have been avoided by just limiting the process to get the human labels alone.

The works that tried to model coherence in text decided coherence based on entities and lexicon relationships. All the works again had separate processing to identify the entities and lexicons first and as it was discussed earlier many subsequent works came just in improving the entity identification process from the base entity grid which proves the challenges in entity detection process. Coherence measurement was not only dependent on the measuring the relevance between these entities but also dependent on the performance of entity identification which becomes an pre-

processing step, thereby causing a bottleneck in all the discussed works (Louis et al., 2012; Elsner et al., 2011; Feng et al.,2012).

Similarly, coherence assessment by means of lexical relationships where the co-occurrence of words, lexical chains are to be estimated first on right measures before they can be used to assess the coherence of the text (Somasundaran et al., 2014; Muyu et al., 2015). The implicit presence of semantic relationship between a request and a response by means of embeddings is a reliable option instead of entities and lexical relationships to assess if a conversation is confused or not.

Works that used embeddings at different levels had RNN and CNN to explore the semantics and syntactics of coherence in text. These works are superior to the previous ones as there are no required huge pre-processing steps of entity and lexical relationship learning and identification (Li et al., 2014; Li, J., & Jurafsky, D., 2017; Dong et al., 2017). These embedding-based works to measure coherence still failed to capture the underlying semantics between a coherent and in-coherent conversation/text which is the nature of similarity and dissimilarity measures in a coherent or in-coherent text. The embedding based works either used pre-trained embeddings or learnt embedding from scratch.

The embedding based works had a straight coherence classifier based on a sentence or sentences concatenated together where the system learnt the semantics from the start of the sentence till the end, look for any discontinuity in a particular semantic pattern meaning presence of words in the text that disturb the pattern obtained in the previous step or word. Such disturbances or changes in the pattern denote the presence of irrelevant words in the text, proving the overall text to be in-coherent. The semantic patterns are represented by embedding vectors. These vectors with the explained semantics along with the target labels are used to train a coherence classifier in case of learning embeddings from the scratch.

Most of the existing embedding based coherent models use the stated approach to learn coherent text/conversation based embeddings except ConveRT by Henderson et al., (2019) which achieves state of art performance in relevant response selection and universal sentence encoder by Daniel et al., (2018) use these semantics in a different way by parallelly learning the semantic embeddings for two sentences, letting the model to finally understand the difference between these semantics for a coherent pair

by placing them closer and in-coherent sentence pairs by placing them far apart in an embedding space. Not just using the similarity/dissimilarity metrics such as distance between the embeddings for the downstream tasks like response selection and quality measurement, providing them as essential, centric features in the neural network while training embeddings, will power these embeddings to learn more semantics in the context of two aspects. Placement of confusion causing irrelevant sentence pairs far apart and coherent/relevant pairs close together.

3. DESIGN AND METHODOLOGY

This chapter talks about the blueprint and the strategy of the entire design to be implemented in the research. It starts with the research methodology adopted, effect of data preparation techniques on the dataset, Components involved in the proposed neural architecture design, different variants of the base design to be experimented and process of evaluation through stages with the right evaluation metric. All the above sections are covered in their theoretical and planning perspective.

3.1 Methodology

The research follows Cross-industry process for data mining (CRISP-DM) framework (Shearer, C., 2000). It includes the sequential stages of Business understanding, Data Understanding, Data preparation, Modelling and Evaluation as seen in the Figure 3.1. One essential process that should be considered during the modelling stage is going back and forth to Data preparation stage. This means that when the modelling does not go in a desired direction, struggles due to improper data such as missing data, imbalanced data, insufficient data then one should revert back to the data preparation stage, collect more balanced and clean data and remodel it to achieve better results in the evaluation stage. The same applies to the evaluation stage when one realises that the evaluation does not meet the business problem, one should re-iterate on the complete process right from data understanding again. This reiteration strategy is followed in the research which is identified and explained in the respective sections.

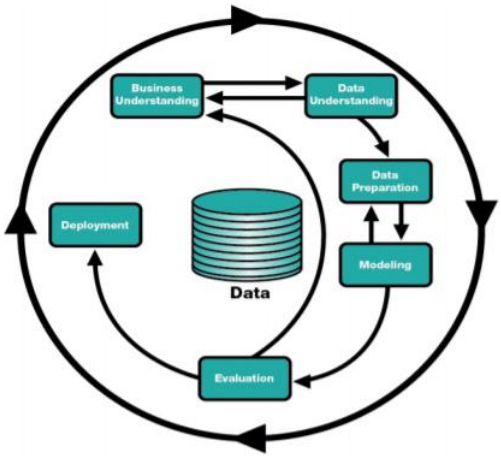

Figure 3.1 Representation of the CRISP-DM model for data analysis

(Shearer, C., 2000)

3.2 Experimental Overview

The whole experimental process undertaken is visualised as subsequent steps in the below Figure 3.2. Experiment starts with the step of labelling data followed by data preparation steps in NLP such as stop words removal, Lemmatization, vocabulary construction, Tokenisation, Trimming, Padding and data splitting. Modelling process followed the data preparation step where the proposed model is designed, and evaluation is carried out for the model's performance against the baseline performance to accept or reject the defined hypothesis.

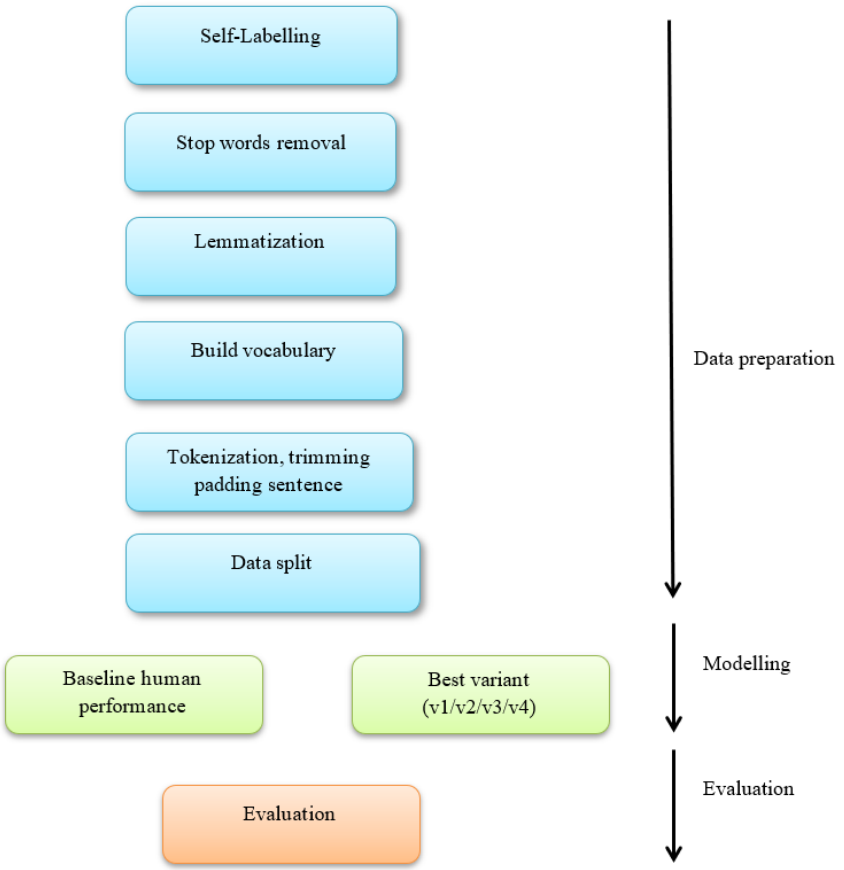

Figure 3.2 Experimental Overview of the process associated with CRISP-DM stages on the right and the NLP, modelling, and evaluation steps illustrated as blocks in the chain.

3.3 Data Understanding

3.3.1 Dataset

ConvAI (Logacheva et al., 2018) is a real conversation-based dataset collected at the Conversational Intelligence Challenge which took place in July 2017. It consists of a total of 4750 dialogues which were conversed between 10 chatbots and 500 volunteers, out of which 4224 are human to bot dialogues and 526 are human to human conversations. The complete distribution is shown in Figure 3.3. The dataset consists of both single turn and multi turn conversations. In case of multi turn conversation, an average of 10.5 utterances occurred per dialogue. Every utterance had an average of 7.1 words as witnessed in the Figure 3.4.

	All dialogues	Human-to-bot	Human-to-human
Total number of dialogues	2,778	2,337	441
Empty dialogues	119 (4.3%)	102 (4.4%)	17 (3.9%)
One-sided dialogues*	560 (20.2%)	520 (22.3%)	40 (9.1%)
Long dialogues**	1719 (61.9%)	1409 (60.3%)	310 (70.3%)
Utterances per dialogue	10.95	10.73	12.13
Words per utterance	7.31	7.48	6.51
Characters per utterance	32.54	33.60	27.56
Unique words per dialogue	45.72	45.00	49.58

Figure 3.3 Complete statistics of the ConvAI.io dataset

(Logacheva et al., 2018)

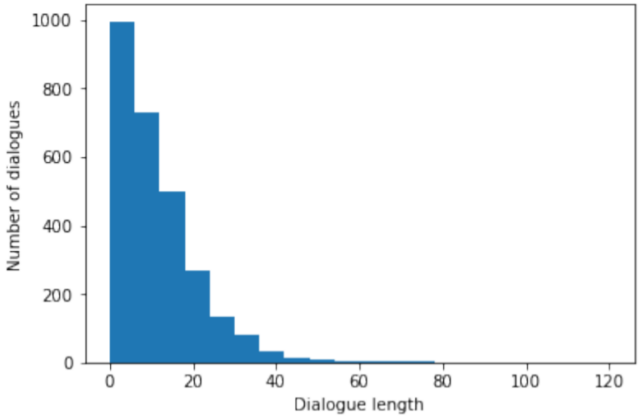

Figure 3.4 Graphical visualisation of the lenth of the dialogues in conversation

(Logacheva et al., 2018)

From the entire dataset, only human to bot, single turn conversations were chosen for the research. These human to bot conversations deal significantly with question-answer type conversations. Out of 2337 human to bot dialogues, 800 single turn conversation dialogues are chosen. Some guidelines are followed during the selection of these dialogues based upon some criteria which are discussed in the next section.

3.4 Data Preparation

3.4.1 Dataset Labelling

The existing datasets which are of conversational type, used to train conversational agents including **ConvAI.io,** do not include any attribute stating whether the conversation is confused or not. So, the picked conversation instances need to be self-labelled with respect to confusion.

The conversations are picked from the entire dataset in a way where the final collection ended up with nearly balanced number of confused and not confused conversation. This would eliminate the process of data balancing using various techniques in the data preparation stage. Also, other guidelines such as avoiding the unclear statements, avoiding very lengthy sentences, removal of expression based words such as alas, wow, hurray etc, multiple punctuation marks, special characters from the conversations, avoiding high complex/vocabulary sentences were followed as during selection of sentences from the dataset to label. Labelling of these carefully picked conversations is done to achieve a dataset in a desired schema of having three attributes, request, response, and 'confusion' as the target label. Request will contain the human statements or questions. Response will contain the bot's response to the human request. Confusion attribute will contain wither of two values, 'confusion' or 'no confusion' which are self-marked labels. The following logical pattern was followed consistently throughout the process of self-labelling.

As stated earlier, A human is left with confusion in a conversation, when the participant receives an answer or response which is irrelevant, or which does not make sense with respect to the question asked. Same way, the confusions in human-bot conversation is identified by the existence of tokens in response statement, which is totally irrelevant to the context words in the request statement. The words in the response statement will be either irrelevant to the actual entity in the request statement

or does contain confusable words. In the dataset, actual request is from human and conversational agent will respond according to that.

For instance,

Request (human): How do you bake a cake?

Response (conversational agent): Just got back from a day of fishing!

In the above-mentioned example, the human participant is left with confusion. The request is about baking a cake and the conversational agent has responded something about fishing. In request statement, the context word is **'cake'** whereas the token **'fishing'** in response statement is nowhere related to the actual request. Thus, this is considered to be confusion and therefore the request-response pair is self-labelled as 'confusion'. Another example,

Request (human): Do you like pizza?

Response (conversational agent): Yes, I like mushroom pizza

In the above example, the context words in both the statements are relevant, hence it is self-labelled as 'no confusion'. Initially only 200 samples are picked from the dataset, self-labelled and subjected to modelling. Later, based upon the modelling requirements and progressive changes seen in the modelling results as data size increased, the number was increased gradually to 800. This increase again happened in three stages which resulted in 400, 600 and finally 800.

3.4.2 Consistency check on Dataset labelling

As the dataset is self-labelled based on the patterns as mentioned in the earlier section, ensuring on the consistency and the quality of the labelling process is really important to make it reliable and trust worthy to proceed on modelling with such labels. Because these target labels are used in the modelling process of training and testing the model, it should of gold standard. Hence the quality of the labelling done is cross verified by having another human annotator label the same dataset. This external annotator is properly instructed on the nature, volume and attributes of the dataset and the objective of the experiment. This annotator labelled the entire dataset as confusion or no confusion. This labelling process is purely done by the individual subjected to one's

knowledge, consent, and semantics as what a particular person considers as relevant (not confusion) and not relevant (confusion) responses.

Now the consistency across the both the labelling process is measured by Cohen-kappa score. It is a test statistic which measures the level of agreement or consistency between the annotators. The kappa result can be interpreted in such a way that it's value of lea than or equal to zero means there is no agreement, or the labelling process is inconsistent across the annotators. Its value of 0.01 to 0.2 indicates a slight agreement, 0.21 to 0.4 indicated fair or decent agreement, 0.4 to 0.6 as moderate consistency and anything above 0.6 is of good consistency and values above 0.8 to 1 are of perfect consistent and agreement. The cohen kappa score is calculated and turns out to be **0.83**, proving perfect consistency.

The resultant kappa score ensures the consistency of both the labelling process by different annotators. But the self-labelled data will be the gold standard data against which the all the experiments train and test the models. This self-labelled data is used to test not only the performance of these models but also used to test the performance of another annotator. Hence the alternate annotator serves two purposes in the research, one to check the consistency and trust worthiness of the self-labelling process and another purpose is to act as the baseline performance of the research in addition to the gold standard data.

3.4.3 Noise Reduction

Data pre-processing is done to shape the data in such a way that it is more consistent and compatible to the model. Noise in the data should be reduced to have only semantic, unique words in the dataset (Furlan, B., Batanović, V., & Nikolić, B. , 2013). These words should only contain the essential entities and the words closely related to those entities. General natural language approaches are used in a sequence of steps. First, the stop words are removed. Words like I, me, my, me, his, he, her, the, a, of etc., do not play the role of entities and related words, but simply add noise. These words do occur so frequently in almost every input sentence. Using NLTK python library, these stop words are completely removed from the all the input sentences, both request and response. Next, all the words in the sentences are changes to lower case for

consistency. The sentences are completely stripped off, to remove unwanted blank spaces.

The total number of words including both the request and response is calculated which is 10736. In the process of removing the stop words from the dataset, 180 pre-defined stop words are used from nltk-python package. As a result of comparing this list with our dataset of 10736 words, a total of 4798 words were removed from both the request and response sequences. The remaining 5938 words present in the sequences are now noise free.

Using the techniques of lemmatization and stemming (Gharatkar, S., Ingle, A., Naik, T., & Save, A., 2017, pp. 1-4), all the words are transformed to their root form. For example, stemming converts the word ‘flying’ to its actual root entity, ‘fly’. lemmatization converts all the different forms of the same word to its root form. A lemma is a citation form of set of words. For example, fly, flying, flew are converted to its root word, ‘fly’. At the end of all these implications, the dataset gets rid of all the words which provide very little or no semantic information with respect to the actual core context or meaning of the entire dataset. The dataset sentences become more consistent, where all the 5938 words are only the entity and related words which form the part of the dataset vocabulary.

3.4.4 Vocabulary

The number and the list of unique words are obtained from the dataset’s request and response sentences. This list of unique words will have reduced noise as a result of the pre-processing them in the previous stage. These unique words form the vocabulary of the dataset. It is also called as dictionary (Furlan, B., Batanović, V., & Nikolić, B. , 2013).

A set of unique words is generated which extracts 1627 words from the total of 5938 words. This is going to be the size of the dictionary. Another analysis shows that removal of stop words in the previous step reduced the actual vocabulary size from 1703 to 1627. Hence a total of unique 76 stop words are present in the complete dataset.

3.4.5 Tokenisation

Tokenisation is the process of converting an input sentence into a list of separate tokens where each word represents a token and the words are identified by splitting the sentence based on blank space (Webster, J. J., & Kit, C. , 1992). For example, the sentence in the dataset, “What is your father” will be represented as “what” “is” “your” “father”. These tokens are assigned with some unique numbers ranging from 1 to the size of the vocabulary (1627) which would make the same above sample sentence appear as ‘2 50 35 200’ where each number is nothing but the index of the tokenised word in the vocabulary. This tokenisation is done using the tokenizer package of keras. Tokenizer object accepts some parameters such as num_words, filters etc. ‘num_words’ is the number of unique words that are present in all the input sentences of the dataset, is assigned with the size of the vocabulary (1627) obtained in the previous step. ‘filter’ is another parameter where a regular expression can be given to filter characters on a pattern. This regular expression is used to filter out all the characters other than numbers and alphabets. This would eliminate all the punctuations, special characters, and symbols from the dataset. This would further reduce the noise in the input sentences.

The table snip as in Figure 3.5 is formed by splitting every sentence, both request and response into individual words and aggregating the result set using count operation on unique set of words derived. This table has only the vocabulary which means the core entity words in the lemmatised form excluding the stop and non-entity words. The word ‘love’ is the most occurred word in the request and response which is followed by the words, ‘work’, ‘favourite’, ’dog’, ‘living’, ‘play’ etc. Only the top 10 recurring words are listed to learn the top influencers of the dataset.

	word	frequency
0	love	116
1	work	71
2	favorite	51
3	kind	42
4	year	42
5	movie	38
6	food	37
7	music	36
8	live	35
9	time	35
10	dog	34

Figure 3.5 Top 10 word Frequencies in the vocabulary

3.4.6 Trimming

The input sentences of the dataset will have different number of tokens. It is expected to make all the sentences look uniform with same number of tokens for consistency. Hence a maximum number of tokens that every sentence should contain is determined. A function is declared which loops through all the request-response sentences of the dataset to find the number of tokens in every sentence as a column. The number of words in every sentence range from 2 to 46. As in the Figure 3.6, the first histogram is plotted with the number of words as values ranging from 2 to 46, to include distribution of words in all the sentence types, ie shortest to longest. It can be clearly witnessed that sentences having more than 10 words are few. Also, most of the sentences have around 5 to 10 words and the sentences having more than 20 words see to be outliers.

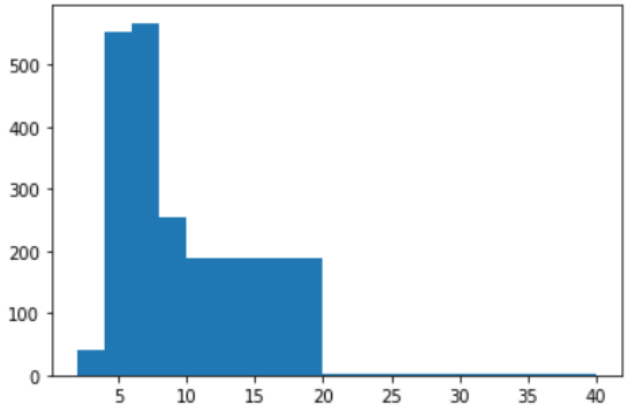

Figure 3.6 Distribution of sentences having from number of words from 2 to 46

To see the distribution within 5 to 20 words range, another histogram is plotted with bin width as 2 as in Figure 3.7. It can be witnessed that most of the sentences have number of words in the range 5-8 and the sentences that have words more than 12 get added to the outliers list. From the two plots, it is clear that sentences having number of words in the range 11-46 are negligible. Out of total sentences of 1600, including both request and response, 1500 sentences have less than 11 words. This means that more than 95% of the sentences has number of words less than 11. Hence the optimal uniform number of words to be fixed in every sentence is set as 10 ie, the variable 'maximum length'. All the sentences are trimmed to maintain the number of words as 10.

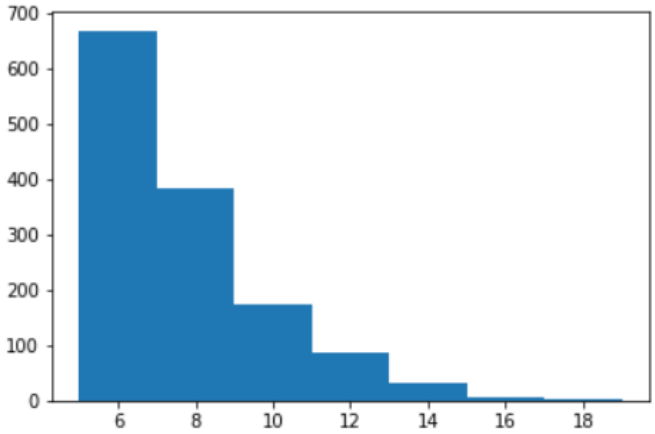

Figure 3.7 Distribution of sentences having number of words from 5 to 20

3.4.7 Padding

The sentences, having less than 10 words which result to be of 1414 strength, are padded with zeros at the right till the number of words in the sentence become 10. Padding is of two types, pre-padding, zeros are left padded (front) to the sentence and post-padding, zeros are right padded (back) to the sentence. It is recommended to follow post-padding as to maintain the meaning of the sentence at the start and the plain zeros at the last All these sentences are transformed to lower case and the punctuations are cleared.

3.5 Data Balance Check

At this stage it is known that the target variable is ‘confusion’ and it is converted into binary class, where 1 represents confusion, the positive class and 0 represents ‘no confusion’, the negative class. The research objective is detecting confusion which is why presence of confusion is treated as the positive class. A Bar plot is done comparing the counts of positive and negative samples,

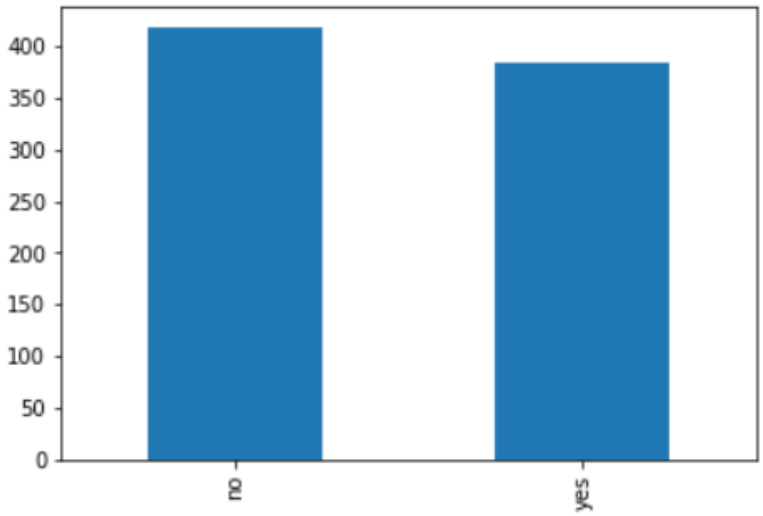

Figure 3.8 Positive and negative class balance test

As seen in figure 3.8, At the first sight, it is witnessed that the data is nearly balanced with respect to its target categories. Number of positive (confusion) and negative (no confusion) samples are in the ratio, 384:418 making nearly 50 % each. This small difference can be tolerated as the model sees enough number of both the classes during its training. Hence, data balance check resulted positive and reduced the further efforts of applying SMOTE, Over sampling of minor class in case of imbalanced data. This also attributes to the reason of proper and cautious sampling of samples during the process of custom labelling and picking samples which was discussed earlier, where the issue of class imbalance, having less number of ‘confusion’ samples is identified during the initial experimental evaluation phase and went back to the stage of business and data understanding to collect more samples of particular class (yes-confusion) to solve the data imbalance issue. This is another example of implication of CRISP-DM methodology in our research which stresses that reverting back to the stages of data collection and understanding is always a better solution to improvise the solution of the

research problem instead of doing adjustments like SMOTE, over sampling etc, in case of the data imbalance issue.

3.6 Data Splitting

Once the data preparation is done, the dataset needs to be split as training, validation, and test partitions. Training set is the proportion of dataset which fits the model. It is the dataset portion on which the actual training happens to reach the point of global minima. Validation set is proportion of the dataset that is used to evaluate the effectiveness of different experiments on the model that is trained using training set. It plays an important role in the hyper parameter tuning process to check the impact of experimenting every hyper parameter combination while training the model. Test set is the proportion of the model to finally evaluate the model's fit. It is used to evaluate the model only after the model is finalised with all its best configurations and ready to be evaluated finally. This portion of the dataset is used as a representative of all the new data that the model has not seen to make sure the model has generalisation capacity (Witten, I. H., & Frank, E., 2002).

As per the split ratio of 8:1:1::train:test:validate, the data partitions have number of samples as ,

Training - 640 Validation - 80 Testing - 80

As told earlier, this split is done only for model training and validation during the experimental phase. For the true evaluation of the model k-fold cross validation is used which is discussed in detail in 'Performance Evaluation' section. k is set as 10, then data split partitions look like,

Training - 720 Testing - 80

Here the dataset is split into 10 partitions, every 10% of data which is 80 samples, denotes a partition. A total of ten evaluation iterations take place where training portion will have 9 (k-1) partitions, 720 samples (75% data) and testing portion has one partition, 80 samples (25% data). Through the evaluation that occurs via 10 iterations, all the 10 partitions, one on every iteration will show up as the test split.

3.7 Modelling

After the pre-processing stage, A Base line model is created as per the hypothesis definition. This Baseline model is considered to be the state-of-the-art performant of this research. The results of the baseline model are the benchmarks for the further experiments. The experiments of 4 variants of the proposed architecture are conducted, tested for the individual's performance against the human's score, the best performant is identified among the variants and compared against the baseline performance results. These experiments are well researched and carried out in the direction of solving the hypothesis to meet the benchmark results.

3.7.1 Baseline Model

Baseline performance set here is human evaluation. Two Real humans have decided upon whether the given single turn conversation is coherent or a confused one. Since the entire research is around detecting the confusion that occurs with a human participant when one interacts with a dialogue system, the baseline performance is set based upon real human scoring. It is also a non-deniable fact that achieving a human comparable performance is always a state of the art in any research. An unbiased human is always going to spot confusion far better than any machine or algorithm as confusion is a personalised feeling, felt by humans as like sentiments in case of sentimental classification.

As per the hypothesis definition, experiments are undertaken to check if the results of the experiments seem to be comparable against the real human accuracy and not exactly achieving equal or greater performance than humans which is practically impossible with the kind of problem statement which is dealt here i.e., confusion detection. As discussed in the data preparation section, the primary annotator is the self-person doing the research. Another human is an external annotator who is made sure to label the dataset on unbiased conditions. These humans' scoring will be the baseline performance against which the developed models are compared and evaluated.

3.7.2 Proposed Design

A Recurrent neural network is modelled as the base artificial neural network type for all the proposed experiments. Python Tensorflow and Keras package are used to carry out the experiments technically. A total of four experiments are carried out as a part of the proposed design. These four variants have the common neural architecture with respect to the following components in a sequence.

1. Embedding
2. Dual Encoder
3. Combiner
4. Feed forward & Activation

Embedding layer – It is the game changer for every experiment. As the complete research problem revolves around the semantic embedding to decide the confusion in a conversation, different experiments are carried out, with only major changes to this layer, to result in efficient embeddings that purely understand the semantical differences between a confused and a clear conversation. The relevance between the request and response, which decides the confusion, is measured by the semantic embeddings. In a confused conversation, the response sequence, rendered by the agent will have entities which are very less or not relevant to the entities present in the actual request sequence. In terms of embeddings, the vectors that represent the request and response sequence entities will be far apart in the high dimensional embedding space for a confused conversation, whereas a non-confused relevant responses' vector embedding would be closely placed in the embedding space.

Embedding layer, when the word embeddings are trained from scratch, the model is expected to plot the high dimensional embedding space, place the relevant and non-relevant entities at right distances apart. The model gets fed about these learnings with the help of training labels confusion (not relevant) or 'no confusion'(relevant), where confusion instances are placed apart. This is similar to training embedding space with methodologies like, Skip gram (Mikolov, T., Chen, K., Corrado, G., & Dean, J. , 2013), negative sampling (Goldberg, Y., & Levy, O. , 2014), GLOVE (Pennington et al., 2014) where, a target label is created for every word to train the model to distinguish the related and non-related words.

It is an inevitable fact that training an embedding layer from scratch with a less volume dataset as in the case, may not end up in a well-trained embedding space. Pre-trained embeddings like BERT (Devlin et al., 2018) and GLOVE (Pennington et al., 2014), ConveRT (Henderson et al., 2019), have been trained with these semantics into consideration. Hence in addition to the proposed embedding layer design, it also uses these pretrained embeddings, make use of the well-established relationships between the entities, fine tune them further with the base architecture and make better classification. Four different experiments are conducted having changes to the embedding layer, such as self-trained embedding, using pre-trained embeddings such as BERT, GLOVE and ConveRT. These four variants are experimented to check the best performant and carry out further analysis and comparison against the base line performance using the best performing variant.

Dual Encoder – A Seq2Seq type (Sutskever, I., Vinyals, O., & Le, Q. V., 2014); (Cho et al., 2014), also called many-to-many recurrent neural network usually include a encoder-decoder architecture, which involves two sequences, an input sequence is given to generate an output sequence. Seq2Vec, also called as many to one type neural network is another type of recurrent neural network where an input sequence is used to generate a vector as an output. A hybrid architecture, which involves all the three components, two sequences and a vector need to be designed. The two sequences belong to the input side which are request and the response sentences. The vector belongs to the output side which is the target label, confusion or no confusion.

The architecture will involve two encoders, one for request sentence and another for a response sentence and is based on the design of ConveRT (Henderson et al., 2019). Another crucial demand exists here, which is parallel and separate functioning of the encoders. Both the request and the response sentences should pass through the encoders separately but on parallel, get converted into respective flattened encoded output vectors. These encoded outputs of both the request and response sequences act like the sentence embeddings for the total request and response. After every word of the sequence gets vectorised through the embedding matrix, get combined through the encoder layers of LSTM units and finally form a single vector representing the whole sentence itself. These sentence embeddings later undergo further processing and finally decide the target label. This further processing starts with a combiner.

Combiner – The Neural network architecture has another important layer following the dual encoder stage which is called the combiner. Individually processed on parallel, the two encoded outputs ought to decide the target label. The concept of combiner is again based on the work of ConveRT (Henderson et al., 2019). where the researchers had a combiner stage which just combines the encoded outputs and calculates the cosine similarity between them. Here instead of concatenating, difference between the vectors in form of distance is obtained. The two encoded outputs will be of same dimensional length representing the consolidated form of all the words of request and response sentences individually.

The combiner stage will calculate the Euclidean distance between the encoded vector outputs of both the request and the response encoders. Euclidean distance between two vectors end up in a single number as it sums up the squared differences between every element of the vectors. But, as we need a vector output from the combiner, distance is calculated between every adjacent element of the vectors and the output vector has the absolute differences between the vector elements. The final summing step is discarded and left as a resultant distance vector itself. This distance vector is the expanded version of the actual Euclidean distance between the embedding vectors. As stated earlier, The training of embedding layer primarily depends upon the request-response pair and the target labels, the combiner stage delivers the distance between the request and response encoded vectors, embedding layers trains the embedding space in such a way that it tries to enlarge the magnitude of this distance vector for a target label confusion and reduces the magnitude in case of 'no confusion'.

Hence, Combiner's Euclidean distance calculation is the important edge that adds to the regular embedding training techniques of skip gram and negative sampling. The single distance vector itself represents three features, one is the encoded representation of the request words, second is the encoded representation of the response words, third is the semantic difference in distance between the request entities and the response entities. These three consolidated input features in form a single distance vector, plus the target label helps in the formation of efficient embeddings. The traditional techniques of skip gram and negative sampling just uses a single feature which is the input word, trains the embedding space based on the target word which will be only relevant to the input word in case of skip gram and both relevant and irrelevant in case of negative sampling. This is an important differentiation and an enhancement that we

add to train semantic embeddings to place the relevant and irrelevant entities at right distances apart in the embedding space.

Combiner stage cannot be implemented directly as there are no straight inbuilt layer packages in keras to just call the Euclidean Distance as a function of the previous layer. Hence it is achieved under a work around which using lambdas and four consecutive layers. Consider A and B are the encoded vectors of request and response respectively,

Layer 1 - Negation = Lambda(lambda x: -x)(B)

Layer 2 - Subtraction = add([A, Negation])

Layer 3 - Squared = Lambda(lambda x: x**2)(Subtraction)

Layer 4 - SquareRoot = Lambda(lambda x: x**1/2)(Squared)

Negation layer does the negation of response vector, Subtraction layer just adds the request vector and the negated response vector which eventually results in Subtraction. Squared and Square root layers do the functionality of just removing the negative signs that would have resulted out of subtraction. As seen, the step of summing up the values before taking square root as in normal Euclidean distance calculation is discarded. Actually, the last two layers can be simply replaced with a single layer of finding absolute value of the vectors which would ignore the negative sign but to match against the euclidean distance formula, the above calculation is maintained.

Feed Forward and Activation – The third phase of our network is a simple feed forward neural network, which process the distance vector through few layers and a final layer having sigmoid activation to decide the confusion or no confusion. The number of layers and units are purely left to the results of experimental iterations in the hyper parameter tuning stage. The objective of having these few feed forward layers serves two purposes. In the variant where the embeddings are trained from scratch, these layers would process the distance vector furthermore through some weights and RELU activations, after its immediate calculation from the combiner before getting injected into last single sigmoid neuron. This would primarily help in improvising the learning of the network from the distance vector rather than having them projected immediately from combiner into a single nonlinear last neuron.

In case of using pre-trained embeddings, the feed forward layers do the fine-tuning part. It is always better to do some fine tuning whenever transfer learning process happens, so that, the pre-trained weights would be adjusted to adapt to the designed network, tuned further according to the input sequences of the dataset and enjoy extra benefits. Absence of these layers, when using pre-trained weights, would make the model less compatible to the new input sequences and just reflect the learnings obtained from the actual sequences that were used in the process of training the original models from where these pre-trained embeddings are obtained.

As in the below Figure 3.9, the whole design of the architecture with all the explained components and layers right from the input layer of request and response till the final output layer of decision on confusion or no confusion, the lower most layer which is the embedding layer primarily changes across the variants to be experimented ranging from training the embeddings from scratch till using the latest state of the art conversational embeddings of conveRT. The respective embedding layers of the different variants are visualised in separate diagrams under respective sections

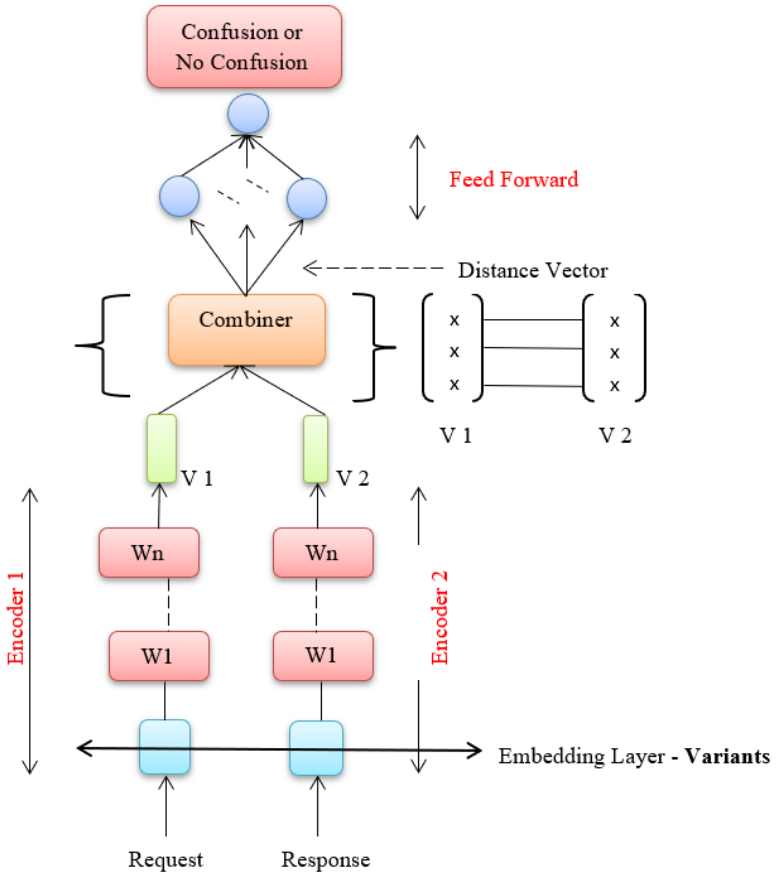

Figure 3.9 Complete Architecture of the proposed base model variant (1)

Apart from the above components, other layers are used in the architecture design. These are introduced in order to solve the challenges that one usually faces while training a neural network to reach the global minima. Positioning and configuration of these layers are decided as a result of experimental iterations in hyper parameter tuning stage.

Regularisation components - As the dataset is of less volume, it is natural that the model would overfit the training data. The proposed design's complex architecture would also add to the overfitting. So, A few drop out layers are included in the network so as to drop certain neurons randomly on each epoch, distribute the learning across all the neurons, control the learning over the training data and generalise the model over the validation and test splits of the dataset (Srivastava, N., 2013). The dropout rates and the position of these drop out layers are decided and restructured through iterations while training the model. Alternatively, L1 (Schmidt, M., Fung, G., & Rosales, R., 2007) and L2 regularisation are also considered to penalise the weights, control the magnitude through some hyper parameters and make them generalised. L1 is not recommended in the variant where embeddings are trained from scratch as reducing the dimensions of a 10-dimensional embedding vector is not needed. L2 Regularisation is used when dropout is not as effective as expected in reducing the overfitting.

Normalization components - There are two normalisation layers to be considered while designing a neural network, Batch normalisation and Layer Normalisation (Liao, Z., & Carneiro, G., 2016). These are generally used to normalise the activations that result out of every neuron so as to avoid computational complexities of higher order activations. This becomes a lot applicable in case of performing calculations on word embedding vectors. Distance is calculated in the combiner stage between the encoded outputs of both request and response sequences, which is nothing but the squared differences of each elements across both the encoded vectors. This would result in higher orders of activations, when passed through further neurons as vector elements are squared. so normalising and reducing their orders is needed to reduce the complex computations in the neurons which have the non-linear activations.

Batch normalisation is recommended in cases of higher batch size, where normalisation happens across the mini batch. As the dataset is of less volume and minibatch will be obviously small, layer normalisation is recommended where the

computation of mean and variance from the inputs to every neuron happens for every training instance instead of once per the mini batch. This would normalise the square root of squared differences that come out of the combiner before being passed into the activation functions.

3.7.3 Self-Trained Embeddings – Variant 1

Word Embeddings of both the request and response are trained from scratch through a embedding layer before the actual encoder component starts. The Embedding matrix is initialised as zeros in the shape (n1,n2), where each embedding for the word is of n2-dimensional length, n1 is the number of unique words in the whole dataset, inclusive of both request and response. Every token or word in the sentence is represented by a vector of n2 numerical elements, which means every row of the embedding matrix represents a token in the dataset. The embedding matrix is trained in such a way that the entities of confused request-response pair are placed far away in the n2-dimensional embedding space, whereas the relevant entities stay closer.

The embedding layer is followed by the separate encoder layers of both request and response sequences. These encoder layers include a LSTM layer, normalisation layer and a dropout layer. LSTM is preferred over SimpleRNN, so that, the LSTM's memory units memorises well on all the entities throughout the sequence. In case of even shorter sequences, usage of LSTM would perform better in case of multiple entities over a simple RNN. As stated earlier, Layer normalisation and drop out layers serve their own purposes. The parallel encoded output vectors which are the consolidated single vector representations of all the words in every request and response sequence pass through the combiner stage, get merged through the distance calculation and fed to the final plain forward layers.

3.7.4 BERT Pre-trained Embeddings – Variant 2

As mentioned earlier, to overcome the limitations of low volume dataset which would not train the embedding matrix well. Bert's sentence embeddings are used to vectorise the request-response pair sequence. BERT is the state-of-the-art language model for NLP (Devlin et al., 2018). It uses Bi-directional, attention-based architecture to learn the contextual relationship between the words.

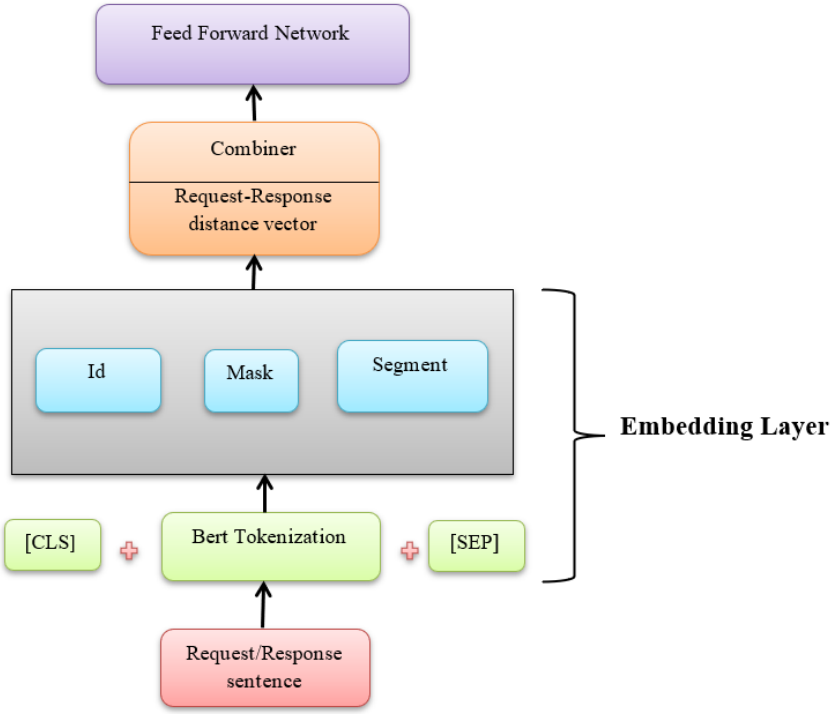

Figure 3.10 Embedding layer of BERT variant (2)

The request and response sentences are looked up against the BERT's model to retrieve the embeddings which are of 768-dimensional length, this replaces the embedding layer with just a look up layer which internally tokenise and vectorise the sentence pairs through stages of BERT tokenisation where every sentence is appended with [SEP] token as a separator in case of two sentences passed and prepended with [CLS] token in case of classification based usage. The next stage being the formation of internal components of ids, masks and segments for every BERT-tokenised sequence as seen in the Figure 3.10, in the embedding layer. This also eliminates the need to pass the embeddings through the encoder because the BERT retrieved embeddings are already sentence level embeddings and do not need to be encoded which is needed in case of word level embeddings.

The retrieved embeddings are projected straight away into the combiner layer for the merging process. As a result of the merging process, the distance vector is obtained between the two 768-dimensional sentence embeddings. The resultant vector pass through a normalisation layer and few more feed forward layers so as to fine tune the model further to generalise for the new input sequences. Also, while initialising the BERT's embeddings, the trainable parameter is set to 'True' in a motive to fine tune

and adjust these embeddings further with the help of distance vector to maximise the distance for confused instances.

3.7.5 GLOVE Pre-trained Embeddings – Variant 3

Another pre-trained word embedding is GLOVE (Pennington et al., 2014), which is based on the semantics of co-occurrence of words in a sentence. This variant is built with an embedding layer, which involves the process of transfer learning the parameters of embeddings from the glove pre-trained vector embeddings. The embedding matrix [n,300] is initialised with the vectors retrieved for every word of the request and response sequence from the glove pretrained dictionary as seen in the Figure 3.11 in the embedding layer. Every word of the sentence gets its 300D embedding vector from the matrix, passes the encoder layers through LSTM units. The encoder also consists of normalisation and dropout layers to avoid higher order computations and overfitting respectively. This is followed by the regular combiner layer where the encoded outputs get merged and goes through rest of the feed forward layers to decide the target label

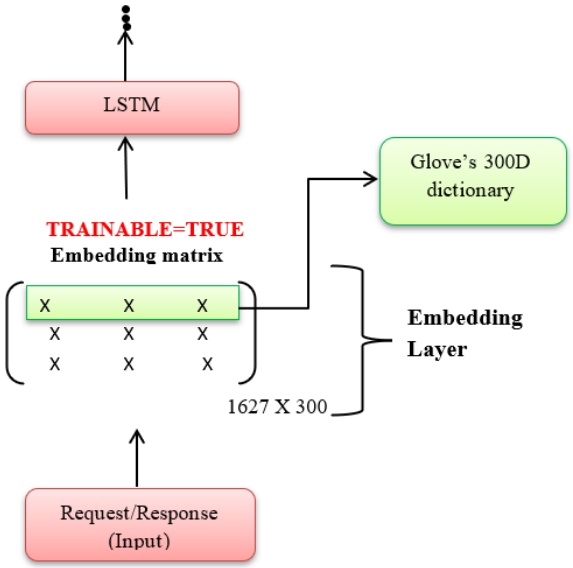

Figure 3.11 Embedding layer of Glove Variant (3)

3.7.6 ConveRT Embeddings – Variant 4

ConveRT is a dual encoder model designed in April 2020 for retrieval type chatbots to select the best response for the request posted. It proved to be the state-of-the-art model in relevant response selection across all conversational agent research works, having the highest response selection score of 72% which is the latest best in the response selection research works (Henderson et al., 2019). ConveRT uses a similar architecture, having 2 encoders one for the request and one for the response and finally generates a similarity score for the request and response.

The original native ConveRT as in Figure 3.12 uses a multi headed attention architecture on the encoders. The encoder is constructed through a series of layers such as sub-word embeddings, where word level embeddings are trained. Since it uses a transformer attention network instead of LSTM or GRU or SimpleRNN to handle long range dependencies, Positional Encodings are used which makes the layers aware on the ordering of words through positions in a sequence. Self-Attention layers actually replace and enhance the work of LSTMs where weightage is given to the passed entity words while dealing with every next token of the sentence. This is followed by feed forward and normalization layers.

The two encoded outputs are finally used to compute the cosine similarity between them. The network also adopted shared parameters technique so that trained parameters on the one encoder are shared across the other in order to reduce the computation, as both the encoders had a homogeneous structure. The model was trained over 10M reddit conversational sentences which included a total of unique 31,476 word tokens.

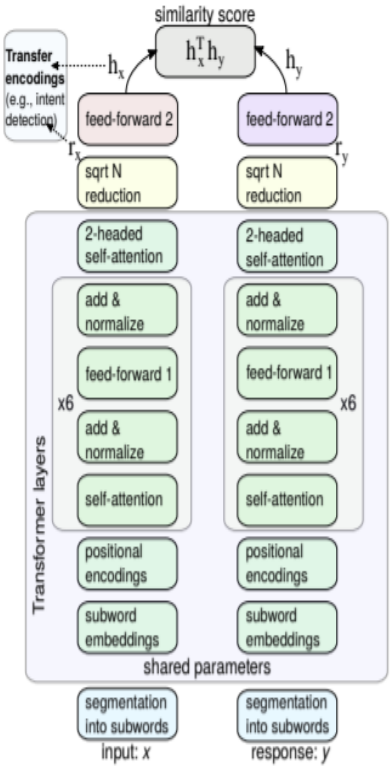

Figure 3.12 ConveRT's native architecture

(Henderson et al., 2019).

Differences between original ConveRT and Variant4 - This research to detect confusion between two sentences, A dual encoder architecture is used as discussed in the Design section. The design is based/similar to the architecture and working of conveRT's model with good differences. The designed network is not a transformer attention-based architecture but uses LSTM layers. Usage of shared parameter mechanism and multi head attentions don't prevail in the constructed design. Another important distinction is the combiner stage which calculates the distance vector which primarily aids the embeddings learning is not present in the ConveRT model. They just calculate the cosine similarity between the encoded outputs and give the similarity score as the output. In overall, the design of this research is a simplified but a different version of conveRT in terms of the concept to study embeddings based on the distance vector.

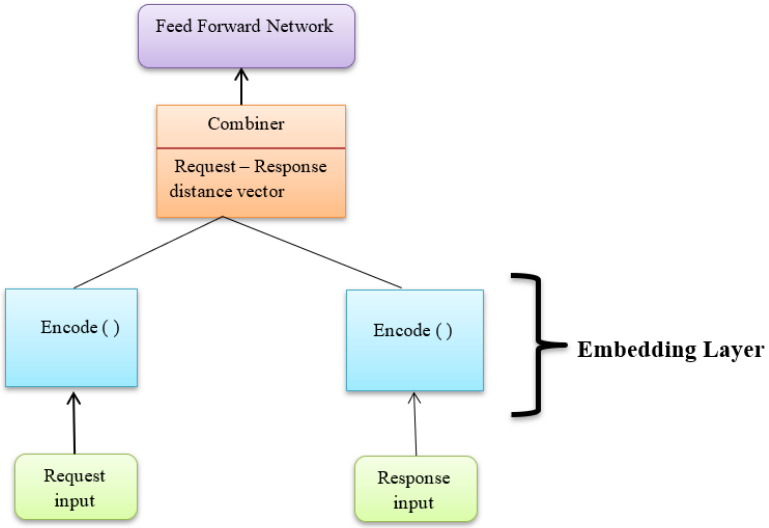

Figure 3.13 Embedding Layer of ConveRT variant (4)

For this variant, the encoded output (1024 dimensional) of the request and response are obtained separately by looking up against the pre-trained word embedding dictionaries of the open source ConveRT model extracts. The researchers had left a python notebook in github [1] which provides the guidance of using their model extract as a look up to retrieve sentence embeddings by passing the entire sentence as a parameter into a function called encode() as in the Figure 3.13 in the embedding layer. This function is initialised through their model extracts. These encoded outputs are actually the processed outputs of all the words of the given sentence. Hence this transfer learning process eliminates the embedding and encoder layers of the proposed model architecture. Once the embeddings of the sentences are obtained, they are merged by the distance calculation and are passed through few layers of feed forward plain neural network.

3.7.7 Hyper-parameter tuning

Running through the discussion of all the stages of the modelling process, a lot of hyper-parameters have been encountered. It is crucial to select proper hyper parameters to increase the model efficiency. Improper usage of these hyper parameters would result in poor results inspite of having a out of the box architecture (Bardenet,

[1] https://github.com/PolyAI-LDN/polyai-models

R., Brendel, M., Kégl, B., & Sebag, M., 2013). Through out the entire process, the collected hyper parameters are Learning rate, Epoch, Mini Batch Size, Patience, L2 Regression, Dropout rate, Feed Forward Layers, Hidden units, Optimiser, Embedding layer size. All the above hyper parameters are decided as a result of tuning process through an iteration of experiments. All the possible values for every hyper parameter are listed and manually tried for each combination of values of different hyper parameters.

Linear scaling increase is followed for certain hyper parameters such as Batch size, Epoch size, Number of layers, Number of hidden units. Logrithmic scaling increase is followed for certain hyper parameters such as Learning rate, L2 Regularisation, Drop out rate as there are infinite possibilities of values in their ranges (Lévesque, J. C., Durand, A., Gagné, C., & Sabourin, R., 2017). An optimization parameter and its direction is chosen. If Validation loss is the optimization parameter, then the direction is to minimise it. If Validation accuracy is chosen as the optimization parameter then the direction is to maximize it. The objective of the hyper parameter tuninf process is to attain the possible extreme direction of the chosen hyper parameter.

In this process, Validation accuracy and maximisation are chosen as optimization parameter and direction respectively. Logical Thinking is put forward to skip few combinations of hyper parameters based on the knowledge of experimenting initial combinations. While modelling the first variant, it would pave a direction to decide on ‘what further combinations’ attempt would result in maximisation of validation accuracy? and skipping which coimbinations would help reduce the iterations logically?’

Hyper parameter tuning is done when actually experimenting and training every discussed variant. No information on the internal configuration of the various models have been discussed so far in this chapter. This configuration includes all the collected hyperparameters. Only the base design and the working intuition of the models have been discussed so far. Even though the base design stays the same, these configuration tends to vary for every model. The complete configuration of every model is decided only when training the models, followed by hyperparameter tuning process. Hence the details of these configuraions are discussed in the next chapter where the real experiments on the proposed design are done and results are laid out.

3.8 Performance Evaluation

There are a lot of performance metrics to be considered while evaluating a model. Based upon the problem statement and the nature of the data, this evaluation metric is chosen. Accuracy is defined as the overall capacity of the model in predicting correctly (Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. , 2001). If the dataset is imbalanced, having majority of positive classes and small number of negative instances, then the model could achieve higher accuracy by simply predicting the positive class all the time. In this case the real performance not only depends on predicting the majority positive classes correctly but also making good predictions on the negative class. Hence Accuracy is not the only metric against which a model shoule be evaluated. Other metrics like recall, precision and F1 score should also be calculated which would measure the model's performance by its prediction capacity on all the classes not just one class. A model which is having higher accuracy, but imbalanced prediction scores across precision, recall and F1 score, then the model cannot be claimed to be good performant.

All the predicted samples fall into four categories, True Positives, which are the samples correctly classified as positive by the model, True Negatives, which are the samples correctly classified as negative by the model, False positives, which are the samples wrongly predicted as positive by the model but are actually negative, False Negatives, which are the samples wrongly predicted as negative by the model but are actually positive. The wrongly predicted samples either fall into either false positives, Type 1 error or false negatives, Type 2 error. Both type 1 and type 2 error decide the error rate. Accuracy deals with only the True positives and True Negatives which identifies the proportion of correctly classified samples from the total samples.

The research objective is to detect confusion, the positive class. The model should be able to capture as much as confusion as it is. It should not miss to spot a confusion in a conversation. It should not classify or predict a sample as negative, ie 'no confusion' when it is actually a confusion, a positive class. This means that the model should try to minimise the false negative portion of its prediction where it misclassifies an actual confusion and miss to spot a confusion in the conversation. Recall is the right evaluation metric to be used when the objective is to minimise fasle negatives (Sarwar, B., Karypis, G., Konstan, J., & Riedl, J., 2000). A higher recall means less false

negatives and the model should be motivated to achieve 100% recall where there are no false negatives, the model spots all the confusions in the conversation, claiming to be at it's best performance.

On the other hand, precision is used when the objective is to reduce the number of false positives (Sarwar, et al., 2000). When there is no clear research objective whether to reduce type1 or type2 error, then f1 score could be considered which is the harmonic mean between the precision and recall (Yang, Y., & Liu, X., 1999). As per the research objective, Recall is considered as the primary evaluation metric to be used to compare the variants and compare the best performing variant against the baseline performance. Accuracy is considered as secondary metric to quantify the general performance of the classifier. As the dataset passed the databalance check, accuracy is trust worthy to be used to evaluate the model's overall perfromance predicting labels correctly. Hence accuracy and recall are calculated for every experiment.

There are two types of validations, Hold-out and k-fold cross validation. K-fold cross validation is always better then than the hold-out validation as it measures the model's generalisation capacity (Yadav, S., & Shukla, S., 2016). A k-fold validation consists of k iterations of training and testing on the same dataset which is k number of hold-out validations. Hence a k-fold cross validation consists of a list of k accuracies, a list of k recalls. For every experiment and variant, a k-fold cross validation is done, having k value as 10, separate lists of accuracy and recall are collected. Also an average of every list is calculated.

The entire model evaluation is done in this research in three stages as in Figure 3.14,

1. **Within a model variant against the gold standard data** – A 10-fold validation is performed against the gold standard data labels obtained by the process of self labelling with which the model is trained. A list of 10 values for recall and accuracy are noted. The average values are calculated. This evaluates the performance of the individual model by comparing the predicted labels to the self-marked labels. Higher average value of the primary metric, recall, ensures the human comparable performance of the model with respect to the gold standard data self-marked labels on detecting confusion.

2. **Across the model variants** – The Average performance score on the calculated metrics of recall of all the variants are compared and the variant having highest average scores for these metrics is considered to be the Best performant of all the experiments. This stage decides the best performant against the gold standard data and one could ensure which variant among all has the highest performance predicting the right labels as that of self-labelled ones with respect to the gold standard data.
3. **Best Performant vs Baseline** – The above two stages of evaluation would decide the models' performance against the gold standard data of self-marked labels. Also, The best performant was decided whose high performance in case, can be claimed to have a human comparable performance. But, to statistically prove the same, two distribution or list of performance scores are needed, one for the baseline human performance and other for the best performant model. Hence in this third stage of evaluation, the labels of the external annotator are compared against the gold standard data labels and a distribution of recall and accuracy are calculated. The distribution of recall represents the base line human performance of both the external annotator and the researcher who self-labelled the dataset because the consistency between the two annotators are proved as strong through higher cohen-kappa value (0.83). Finally the obtained distribution and the best performant's recall score distribution are compared. This comparison is done by means of statistical tests and results are concluded by means of p-values.

At the end of all three evaluation stages, two confirmations are done,

1. If the proposed models' and the best performing model's performance stay comparable to gold standard data having self-marked labels. This is obtained through through stages 1 and 2 which is done for every variant built.
2. If the best performing model's perfromance stay comparable to external annotator human's perfromance. This is obtained through stage 3 which is done once at the end of all experiments.

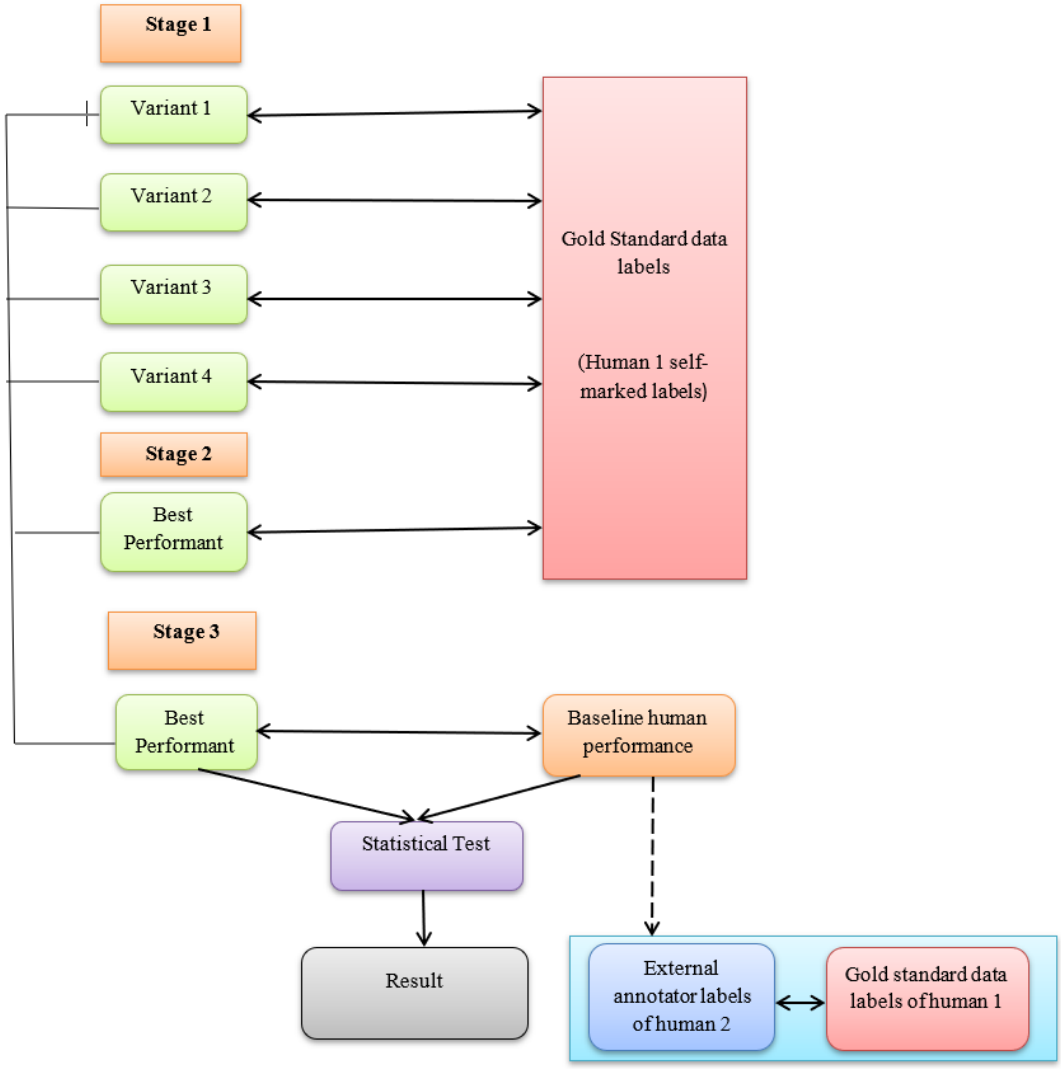

Figure 3.14 Stage 1, Stage 2, Stage 3 of evaulation

The proposed models inturn get tested against two humans' labelling performance which proves for better consistency. Before doing the statistical test of comparison between the derived human distribution scores of recall and the best performant's recall distribution, A normality test is made on these distributions to decide the statistical test. A t-test is a statistical hypothesis test which is used to tell whether two groups are similar or different based on the mean of an attribute that both the groups have (Kim, T. K., 2015). Internally it tries to determine the degree of overlapping between the two groups. There are two types, independent and paired t-test based on whether the comparing groups are independent or dependent on each other. This test is done when the distribution of the numercal attribute which is to be compared for both the groups are determined to have a normal distribution. For non-parametric distribution, Mann-Whitney U test is conducted.

Shapiro-Wilks normality test is used if both the distributions are normal. This test internally makes a null hypothesis that the distribution of the given input is normal at the given alpha level. It results out a p value which when greater than the alpha value, the null hypothesis is rejected, proving that the distribution is not normal for the input. If the p value is less than the given alpha value, then the null hypothesis is accepted, proving that the distribution is normal for the input. The alpha level is set to 0.05 and the input list is passed into the normality function of Shapiro-Wilk test. Here on, for all the normality test, the same procedure is followed.

The chosen statistical test is conducted with confidence level of 95% and a p-value is calculated. This confidence means that a 5% tolerance is accepted. A p-value is the measure of outcome of a hypothesis test which is basically t-test here. Here the two groups are ‘best performing variant’ and the ‘baseline human performance’. The attribute of comparison between these two groups is recall. If the p-value is less than 0.05, it means that the overlapping between the two groups on the comparison attribute is less than the accepted tolerance of 5% and it can be claimed that it is a 95% confident statement that the two groups are different with respect to the chosen numerical attribute. If the p-value is greater than 0.05, then it means that there is overlapping between the groups on the attribute greater than the accepted tolerance of 5%.

This research is motivated to achieve the previous statement to have acceptable overlapping between the ‘best performing variant’ and the ‘baseline experiment’ on recall values. This would tell that one of the proposed experiment’s performance on detecting confusion is comparable or some way equal to that of the baseline performance of real human’s ability to spot confusion in a conversation. Hence the claimed alternate hypothesis of achieving human comparable performance, can be accepted and the null hypothesis can be rejected only if the resultant p-value is greater than 0.05.

4. RESULTS, EVALUATION AND DISCUSSION

This chapter includes the results obtained by implementing and executing the design strategy put forth in the previous chapter. It presents and analyses the results obtained in Baseline performance test, testing the different variants of the model against the gold standard data. Final sections include the comparison and evaluation of the best performing variant against the base line performance and the results to solve the hypothesis followed by overall discussion on the results.

4.1 Baseline performance

Having proven the consistency of self-labelling process against external annotator's labels, it is used for further modelling process. For evaluation stages of 1 and 2, the gold standard data which is self-labelled form the baseline performance against which every individual variant is tested on performance. For evaluation stage 3, the baseline performance results are the outcomes of comparison of external annotator labels against the gold standard data labels which are self-marked ones. Since the entire dataset of 800 samples is labelled by the external annotator, all these 800 labels are compared against the actual gold standard data labels and the evaluation is done. A confusion matrix, as in Figure 4.1, is built to visualise all the classifications and the accuracy, recall scores are calculated.

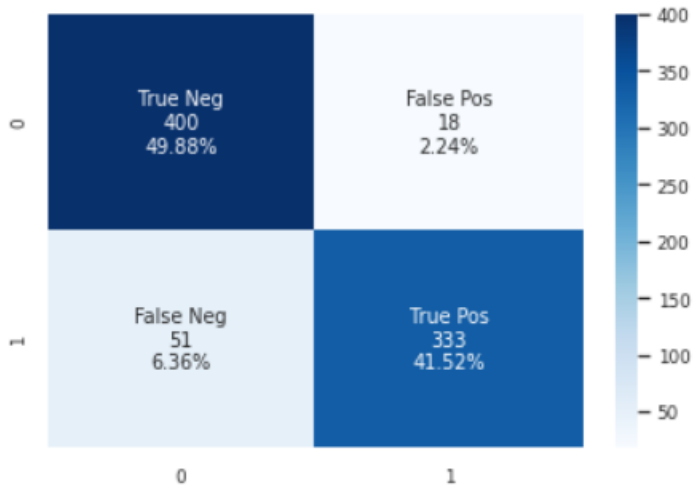

Figure 4.1 Confusion matrix of base line performance

Accuracy	91%
Recall	87%
Type 1 error (FP)	2.24%
Type 2 error (FN)	6.36%

Table 4.1 Baseline performance results on Accuracy, Recall and Error rate

The Proportion of True negatives is the highest which means that the human is good at spotting non-confused sentences, relevant and coherent conversation. 51 confusion spots have been mis-labelled by the human, where they actually failed to spot the incoherent confused conversations. Hence the false negatives, type 2 error make up to 6.36 percentage contribution to the total error rate of 8.6%. The relatively higher proportion of false negatives made the recall score lower compared to that of accuracy. On the other side, Type 1 error contributed 2.24% to the total error rate. As mentioned earlier, the research looks to maximise recall, capture and spot more confusions in conversation. The baseline performance result is seen in table 4.1, The obtained recall score is 87% is comparatively less than that of accuracy - 91%. This forms the baseline performance to be compared against the best performing variant in evaluation stage 3. This is the performance result of evaluation of human labelling for the entire dataset. Once the best performing variant is deduced, a list of 10 accuracy and recall scores for the base line performance is obtained by comparing 10 folded proportions of the annotator's labels against the respective proportion of gold standard data labels in parallel with the best performing variant's 10-fold validation.

4.2 Self-trained Embeddings – Variant 1

A dual encoder recurrent neural network is built as in the design section with the components, Embedding layer, Dual Encoder, Combiner, Feed forward and activation layer. In the embedding layer, an embedding matrix in the shape (1627,10) is initialised where 1627 came from the vocabulary size and 10 is decided to be the size of embedding vector for every token. A parallel, individual, dual encoder component, one for the request and another for the response tokens, with LSTM layer, having 50 hidden units is added as the successive layer. A drop out and recurrent drop-out rate

are chosen as 0.3 and 0.2 respectively. A Normalisation layer is added just after the embedding and before the LSTM starts to get the embedding vectors normalised just before they enter the LSTMs. The combiner component follows the encoder to calculate the distance vector using the request-response encoded outputs. The following feed forward layer has a total of 4 component layers bundled, as Normalisation-Dense with 20 hidden units and activation as RELU-Dropout with rate of 0.3-Dense with 10 units and activation as RELU. A final dense layer with a single hidden unit and a sigmoid activation function closes the network.

Due to a smaller number of 640 training samples, A mini batch size of 20 samples is chosen and the training happens for a total of 50 epochs. As there are 640 training samples, batch size of 20 samples and 50 epochs, technically the gradient of the loss curve changes for every 20 samples of the training set during the training process. There would be a total of 32 steps in the march of the loss function towards the global minimum.

As a part of hyper parameter tuning process, A series of experiments are done in a motive to minimise the validation loss and maximise the validation accuracy. All the above configuration numbers are final choices of hyper parameters used in the experiment iteration resulted out, after trying out different combinations on the below given ranges as in Table 4.2

Hyper Parameter	**Range**	**Final Selection**
Learning rate	0.0001 – 0.1	0.01
Epoch	10-100	50
Mini Batch Size	1-20	10
L2 Regularisation	0.001 – 0.1	0.05
Dropout rate	0.2-0.5	0.2
Feed Forward Layers	2-7	3
Hidden units	10-256	LSTM - 50 Dense1 - 20 Dense2 - 10
Optimiser	ADAM, RMSPROP, SGD	ADAM
Embedding layer size	5-50	10

Table 4.2 Hyper parameter tuning of variant 1: Ranges and Final values

As mentioned earlier, when the values were chosen for the hyperparameters from the mentioned ranges, jump in the value of the hyper parameter happens through either linear or logarithmic scaling depending of the number of possible values within the range of specific hyper parameter.

With the finalised hyperparameters, the training and evaluation of model are done with the training and the validation proportions of the dataset, respectively. During training, the movement of training and validation accuracy through the 50 epochs can be visualised as below.

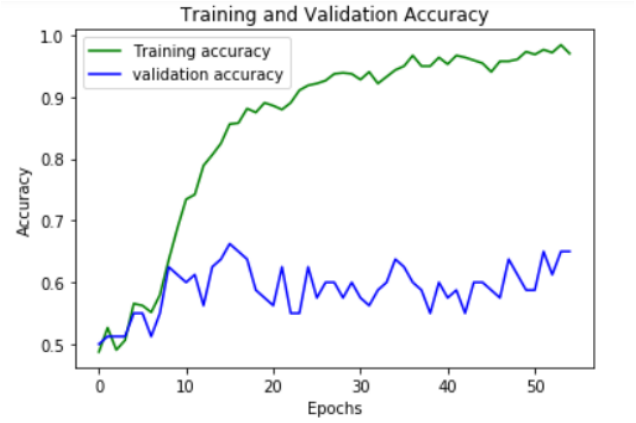

Figure 4.2 Training and Validation Accuracy progress through epochs – Variant 1

Figure 4.3 Training and Validation Recall progress through epochs – Variant 1

As it can be witnessed in Figure 4.2, the accuracy of the validation gets wiggled a lot since the first few epochs itself, while the training set kept gradually improving the model's learning, reaching its maximum of 97%, whereas, the validation accuracy

reached a maximum of 62% after lots of ups and down. The ability of the model of not to miss spotting confusions, gained through epochs can be witnessed by the recall graph as in the Figure 4.3. It is better as compared to the model's general accuracy of predicting the right labels, has the similar pattern as that of accuracy through the epochs and ended up in 65%.

Having the training process completed, the model is evaluated through 10-fold cross validation which resulted in a list of 10 accuracy scores and recall scores as in the below Table 4.3.

	List of 10 Scores	**Average**
Accuracy	[58.75, 57.5, 58.75, 51.25, 56.25, 56.25, 63.75, 53.75, 57.5, 52.5]	56.5
Recall	[54.05, 66.67, 60.9, 75.0, 58.24, 60.91, 65.11, 52.56, 55.15, 53.66]	58.47

Table 4.3 Accuracy and Recall obtained from 10-fold cross validation – Variant 1

This self-trained embedding variant type ended up in low values of recall and accuracy obtained when compared against human labels. This is mainly because the embedding layer initialises a embedding matrix which has a total of 13840 parameters to get trained. Since it starts with just zeros for all the points in the matrix, gradually learns from the combiner's distance vector and the target labels, it becomes so challenging for the model to completely derive all the 13840 elements of the embedding matrix from scratch through semantically related points through a dataset which has only 800 instances. Hence variant 1 does not have a human comparable performance.

Notable Challenges faced and solutions framed during this experiment is described in detail in the following paragraphs,

Dimension of Embedding layer -A hurdle in obtaining higher validation accuracy was the decision on the size of the embedding vector. Initially the size of embedding was chosen as 100. Later during the training process, it was realised that the model was not able to learn semantic embeddings with such dimensional size. The root reason is the less volume dataset which makes the text corpus size low. Hence after reducing the

embedding size there was a reasonable progress in the validation accuracy. The reason is that with such a less volume dataset, the model struggles to plot a 100 dimensional embedding space to place every word by learning the semantics across all the dimensions and when the size was reduced it was able to do a better job in plotting every word on just a 10 dimensional space. Small text corpuses will have usage of same words at less number of different instances and situations, the probability of co-occurrence of words become less which would eventually get the model to suffer in distinguishing the related words which frequently co-exist in the corpus and the non-related words that rarely or don't co-exist in the corpus instances. Under this situation, having a bigger embedding space adds to the model's struggle. Another take away from this challenge faced is the usage of pre-trained embeddings over training embeddings from the scratch. As these pretrained embeddings are well trained over huge text corpuses like Wikipedia or any other big text bodies, that has millions of words, they have well, semantically built embedding vectors for every word than the embeddings trained from scratch using small corpuses having just 10,000 words as in this case.

Overfitting – During the training process, there existed a clear overfitting, having a training accuracy and validation accuracy of 99 and 52 respectively. Hence to avoid overfitting, techniques like dropout, L2 regularisation, Reducing the complexity of the model through optimal number of layers and neurons. All these hyper parameter values are tried with values through the mentioned ranges as a part of the hyper parameter tuning process. Even though, there existed overfitting after a lot of adjustments, A better performance was reached as compared to the initial stages. Another important parameter that helped to be better at this issue which was increasing the number of samples collected. This is another instance where the research followed the nuances of CRISP-DM methodology of leaping back to data/Business understanding from the Evaluation stage. To put in detail, initially the total number samples that was collected and custom labelled was 400 where just 320 samples occupied the training set. Training with this number of samples was the root cause of over fitting. When this was realised and went back to the data collection process and collected, labelled furthermore samples which doubled the previous number, it witnessed on the reduced over fitting.

Stuck in local minimum - Another important challenge which was observed was that the model got stuck up in a local minimum. Local optimum is a sub optimal point in the loss curve where the training loss appears to be minimised, gets stuck in a valley on the loss curve and struggles to get out of it to march towards the global minimum which is the actual optimal point on the loss curve. Training loss got stuck at 0.7 and for all the further epochs after 30, same loss value got repeated. Couple of solutions were tried to solve this out. First was increasing the learning rate on a logarithmic scale from 0.001 to 0.01. The assumption was that increasing the step size of the gradient march will help the loss value to break out of the local minimum. But this solution did not solve the local minimum issue, instead resulted in overshooting of the loss values where the training loss increased and decreased on a random pattern.

Second, decreasing the mini batch size from 40 to 10. This solution helped to solve the local minimum issue where the loss started to decrease further from 0.7. The intuition realised behind this solution is that decreasing the batch size makes the path of the gradient steps towards the global minimum, untidy by wiggling so much which causes the gradient to jump out of the sub-optimal valley. Having a bigger batch size, say when the batch size is the entire number of records where the number of batches will be one, the gradient steps will be straight, tidy and does not wiggle/shake enough to break out of the valley.

Again, all these hyperparameters were decided on their values as a part of tuning process. The solutions to all the challenges faced came out to be the results of choosing proper values for the hyper parameters in the hyper parameter tuning process after a lot of iterations which finally resulted in the values as seen in the Table 4.2. Hence the same combination of hyperparameters is to be used as start-up value combinations in all the forth coming experiments as they have been finalised as the best settings to render maximum validation accuracy, after a sequence of experiments. This led to the approach of Bayesian search (Lévesque et al., 2017) instead of random search where the direction of usage of meaningful combination of hyper parameters is found in this first variant modelling and they are chosen wisely for the further experiments as the knowledge gets built on what direction of combination choice should be taken forward and what combination not to try.

4.3 BERT pre-trained embeddings – Variant 2

As it was witnessed that the performance of the self-trained embeddings variant is not so good to be used for the comparison against the baseline model. This is purely because of the less volume dataset. Hence the state-of-art model of NLP, BERT's well trained embeddings are used to represent the words in the request and response sentences. Same architecture as in the previous model is followed but the embedding layer is removed. First the pretrained BERT model is initialised as a function. BERT usually calculates embedding for a sentence and not for individual words. It expects two ordered sentences as input while we use it to retrieve embeddings. When the input sentences, request and response are passed as inputs to the initialised BERT model, the derived components, Id, masks and segments for the input sentence are thrown in the shape [1,8,768] in form of a tuple. The sequence output embedding vector of 768-dimensional size is extracted, used to represent the request/response sentence.

Once the embeddings for the inputs are retrieved, they are fed straight away into the combiner to calculate the distance vector. The feed forward layer has the architecture of Dense-Normalisation layers repeated twice having a total of 4 layers. The first dense layer has 256 hidden units and the next dense layer has 10 hidden units, a similar configuration as the first variant. Configuration settings for epoch is increased to 60 as a result of hyper parameter tuning process specific to this variant. batch size is maintained as 20, same as that of the previous variant.

During training, the movement of training and validation accuracy through the 60 epochs can be visualised as below.

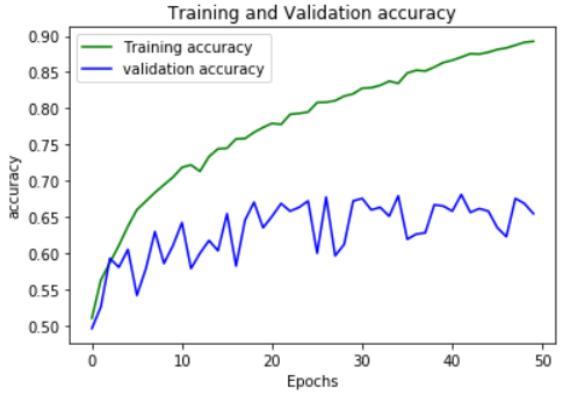

Figure 4.4 Training and validation accuracy progress through epochs – Variant 2

As it is winessed in the above Figure 4.4, the validation accuracy started to grow as that of training accuracy till 4 epochs, later training accuracy had a steady progress but the validation accuracy did not improve at all but sent up and down with very little improvements. Recall growth through epochs, also had a similar pattern. Since the validation accuracy went back and forth, a lot of adjustments are done on the hyperparameters to counter the issue. Initially it seemed to be a overfitting issue through the epochs between 10 to 20, hence different over fitting solvers are attempted such as increasing drop out to 0.6, adding L2 regularisation of 0.05. Later in an attempt to improve the tidyness and compatability of the dataset with respect to BERT's input, the common sentence based index tokenisation is replaced with BERT tokenisation. This did not help on a large scale but pushed the accuracy to 60 range. Changing the optimisers, placement of normalisation layers are some more attempts done to mitigate the issue. Finally the 'trainable' parameter is set to True as to fine tune the bert embeddings to learn futrther based upon the target label and the distance vector, it slown down the training to a great extent as the embedding matrix had 109482241 parameters to be fine tuned and ended up running out of memory. Later setting it to False, the training got completed.

A 10-fold cross validation is done which resulted in a list of 10 accuracy scores and recall scores as seen in the below Table 4.4. This process was a bit challenge for this variant because of memory issues as training the huge BERT for 60 epochs*10 drained the CPU memory.

	List of 10 Scores	**Average**
Accuracy	[70.37, 56.79, 60.0, 63.75, 58.75, 62.5, 71.25, 62.5, 67.5, 58.75]	63.71
Recall	[78.05, 50.0, 64.71, 55.88, 72.97, 64.29, 71.79, 50.0, 76.92, 52.5]	63.21

Table 4.4 Accuracy and Recall obtained from 10-fold cross validation – Variant 2

Though BERT embeddings are originally trained in the native model on huge corpus of 2500 million words, its embeddings do not perform much better than the ones trained using just 10,000 words in the previous variant. Also, it does not seem to be comparable against human performance considering the low recall value.

The challenges faced and the solutions framed during this experiment are described in detail as follows.

The reason for the less effectiveness of BERT could be a glitch in the code with respect to the formation of components such as masks, ids and sengments, needed to use BERT. It is very less flexible to use BERT as a part of the pre-trained embedding layer due to its complex architecture. Every input sentence neeeds to undergo a lot of stages such as BERT tokenisation, creation of masks, ids and segments, conversion of tokens into ids, create positional embeddings, Random masking of the sentence etc obce the BERT model is initialised from th etensor flow hub. Its is less transparent in terms of usage as one could not know if the results at every stage are as expected by the BERT. Usually BERT needs two sentences as inputs. In the experiment, to satisfy the need of grabbing just the embeddings from the BERT model, the second sentence is passed as none. All these added complexities could reasonably lead to some glitches in the code, lead to not a great performance inspite of BERT's being the state of the art performant, it does not tailor filt the actual use case of the experiment.

Higher order Embeddings - When the combiner calculated the distance vector between the request and the response embeddings, the resultant vector had higher orders due to multiple calculations. This resultant higher order embeddings, when passed through the feed forward layers, it resulted in poor training and validation accuracy. When a normalisation layer was added just before the dense layers of feed forward stage, there was a push in the training accuracy from 80 to 90 during the 30-35 epochs.

4.4 Glove pre-trained embeddings – Variant 3

Glove, Another Pre-trained embedding based on co-occurrence of words is used to retrieve embeddings for the request and response sequences. Unlike BERT, it does not provide embeddings for sentences, but for words. Hence an embedding layer is included with an embedding matrix in the shape (1627,300) is initialised where 1627 came from the vocabulary size and 300 is the size of glove embeddings. Following a similar architecture as the self-trained embeddings variant, it has a LSTM layer of 256 hidden units on both the encoders which get merged at the combiner, followed by a sequence of 256 units dense layer, 100 units dense layer and a final 1 unit sigmoid

activation dense layer. This again consists of normalisation layers before every dense layer.

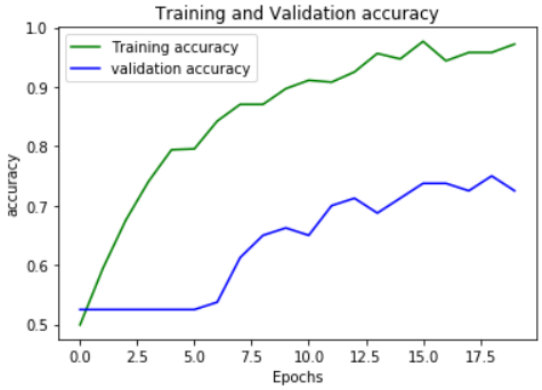

Figure 4.5 Training and validation accuracy progress through epochs – Variant 3

In Figure 4.5, It can be clearly witnessed the progress of the training and validation accuracy on every epoch. The validation accuracy maintained the same value 53% without any improvement till 5 epochs, later got its leap and reached 68% at the end of 10 epochs. There on, it gradually increased with little ups and downs and finally reached a value of 72%

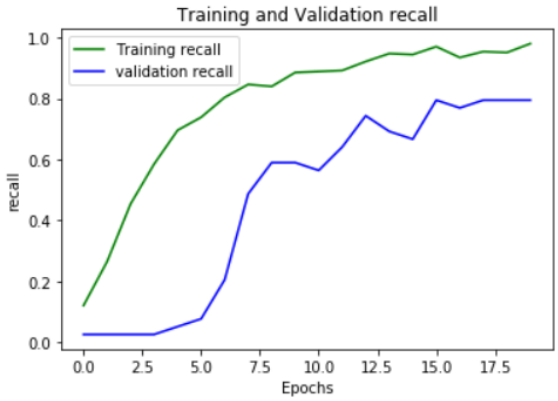

Figure 4.6 Training and validation Recall progress through epochs – Variant 3

Similarly, the recall progress through the epochs stayed very low at its 10% till 4 epochs and slowly picked its pace to reach 60% at its 10^{th} epoch, 70% at its 13^{th} epoch. Later there is a downfall to again 60 % and immiediately leaped to 70 and finally to 79%. There is an irregilar pattern in the progress after 13 epochs. Also last 3 epochs added no improvement to the recall, ehile training recall almost reached 100% which means there are high chances that there will no further progress at all on recall.

A 10-fold cross validation is done which resulted in a list of 10 accuracy scores and recall scores as seen in the below Table 4.5,

	List of 10 Scores	Average
Accuracy	[76.25, 73.75, 70.0, 68.75, 71.25, 66.25, 70.0, 77.5, 72.5, 80.0]	72.625
Recall	[75.68, 73.68, 66.67, 57.78, 64.1, 55.56, 62.5, 82.05, 65.79, 75.68]	68

Table 4.5 Accuracy and Recall from 10-fold cross validation – Variant 3

Glove's 300 embedding trained on a corpus size of 840B tokens outperformed the BERT's performance in the undertaken research. It is clearly witnessed on how the accuracy gets improved as different pre-trained embeddings are used right from using self-trained embeddings. Secondary performance metric, Accuracy is better than the primary metric, recall. Considering accuracy, this variant crossed 70 which is a pretty good performance from a model trained with less instances. The average recall score is 68% which is far better than the previous variants. Also, when took a deeper look into the 10 recall values that resulted in the 10-fold cross validation, 5 test iterations resulted in recall values around 75 which proves this variant to be a good performing one. Though the results are not so attractive enough to deserve a human comparable performance, the experiment resulted in a decent and much better performance from the previous experiments. This results out from the first stage of evaluation.

Challenges faced and solutions framed during this experiment are described in detail as below,

During the training process, when the training accuracy reached 96 %, the validation accuracy reached a maximum of 78 %. For the further epochs through 11, as the training accuracy gradually increased from 96%, the validation accuracy just continued to be the same 78 % for nearly 4 epochs while training accuracy reached its new maximum of 97.8%. Later for the next few epochs through 16, the validation accuracy started dripping off to 76, 75 and so on while the training accuracy moved very slowly

upwards. This indicated over training of the model which started to reduce the validation accuracy. The training of the model should have been stopped when it reached an accuracy of 96% from when the validation accuracy stayed idle and later started to reduce. To mitigate, Early stopping was introduced as a call back function which will be triggered for every epoch. The patience value was set to 5 and monitor to 'val_acc'. This means that when the validation accuracy remains unchanged for 5 continuous epochs, the training should get stopped automatically. This patience value is another hyper parameter which was projected into the hyper parameter tuning process to find its optimal value on a linear scale range. Later this early stopping call back was injected into all the experiments, even for the ones that were done before hand to avoid manual stopping of the training.

4.5 ConveRT pre-trained Embeddings – Variant 4

Final experiment is on the state of art model in response selection, which is conveRT. As mentioned earlier, the architecture of the proposed design is studied and inspired from this conveRT model. This variant has no embedding layer and retrieves 1024 - dimensional sentence embeddings from the conveRT model extracts which are placed in tensorflow hub. A function is initialised which returns the session based 1024 - dimensional vector results for any sentence passed through it. These direct encoded outputs for both request and response encapsulate the embedding and dual encoder layers of our original design and directly start from combiner. A distance vector is computed between the outputs and a simple feed forward neural network is constructed in the pattern, Dense with 256 units – Dropout rate of 0.5 – Dense with 20 units with l2 regularisation of 0.05 – Dense with 10 hidden units – Dense with 1 unit. All the intermediate layers had RELU and the final layer had Sigmoid as the activation.

To check the performance of the original ConveRT embeddings, few sample sentences were passed through the encode function and tested for the comparison of Euclidean distance between the relevant sentence pairs and irrelevant ones. One such example is illustrated below,

Request - "iam hungry now"
Response - "what music you like to hear"

This is a sample pair picked from the collected dataset. This pair has the label as ‘yes’ for confusion. It is obvious that confusion will incur in human if he/she receives such a response from a chatbot. When these sentences were passed through the encode function of the conveRT model extract, two encoded outputs were obtained. As per this variant’s design, distance vector is calculated and sum of the points of the vector is computed, the resulting value is **41.328** units.

The same sample pair again was augmented with right suitable response manually for this test as,

Request - “iam hungry now”
Response - “what food you want to have”

After augmenting the response, the same steps were repeated, and the resulting sum of distance vector is **18.57** units.

It is clear that the value of the confused pair is greater than that of a non-confused pair. These values imply that the first pair of confused sentences are placed far apart to each other and the second pair are placed close to each other in a ConveRT’s trained embedding space. But ConveRT trained word embeddings and not sentence embeddings, so when a closer look is given to the sentences, the entities in the first pair, hungry and music are actually placed far away, the entities in the second pair hungry and food are placed close to each other in the ConveRT’s embedding space. The entity words’ embeddings combined with other words’ embeddings form the encoded vector, representing the whole sentence but the actual logic revolves around the entity words. From this test, it is an inference that the core intuition of using the combiner stage in the proposed experiment, where the distance vector is calculated, then feed forwarded to decide the confusion or no confusion, makes true sense.

Also, a relevance matrix is plotted using heat map which darkens the junction boxes of relevant sentences and lightens the junction box of irrelevant sentences. The correlation or relevance score of 0.3 or greater states that the sentence pairs are related to each other and are relevant. These scores are generated by means of cosine similarity between the ConveRT’s embeddings of the input sentences. As we see, the sentence pairs, “I need thousand Bucks – Why do you need a lot of money” are so correlated with a degree of 0.42. The degree of correlation or relevance between the

sentences is portrayed by the degree of darkness. As the relevance increases, the colour gets darker.

Another interesting example is as follows, consider the input request, ‘What do you do for living’. It’s actual response in the dataset is ‘I work at my parent’s firm’ which looks to be a non-confusing, relevant response for the request. Hence ConveRT’s embeddings had a correlation of 0.42 for this pair. Also, the same request has a relevance of 0.35 with another response ‘I like to do my job’. These sentences actually do not come as a pair in the dataset. Due to the fact that entity words of these two sentences, job and living are related to each other in real sense. Hence their relevance score is close to that of real relevant response. Another similar example will be the sentences ‘what do you do for living’ and ‘why do you need a lot of money’ which have good similarity score of 0.4 as compared to the original pair ‘I need thousand bucks’ – ‘why do you need a lot of money’ which have the score of 0.42.

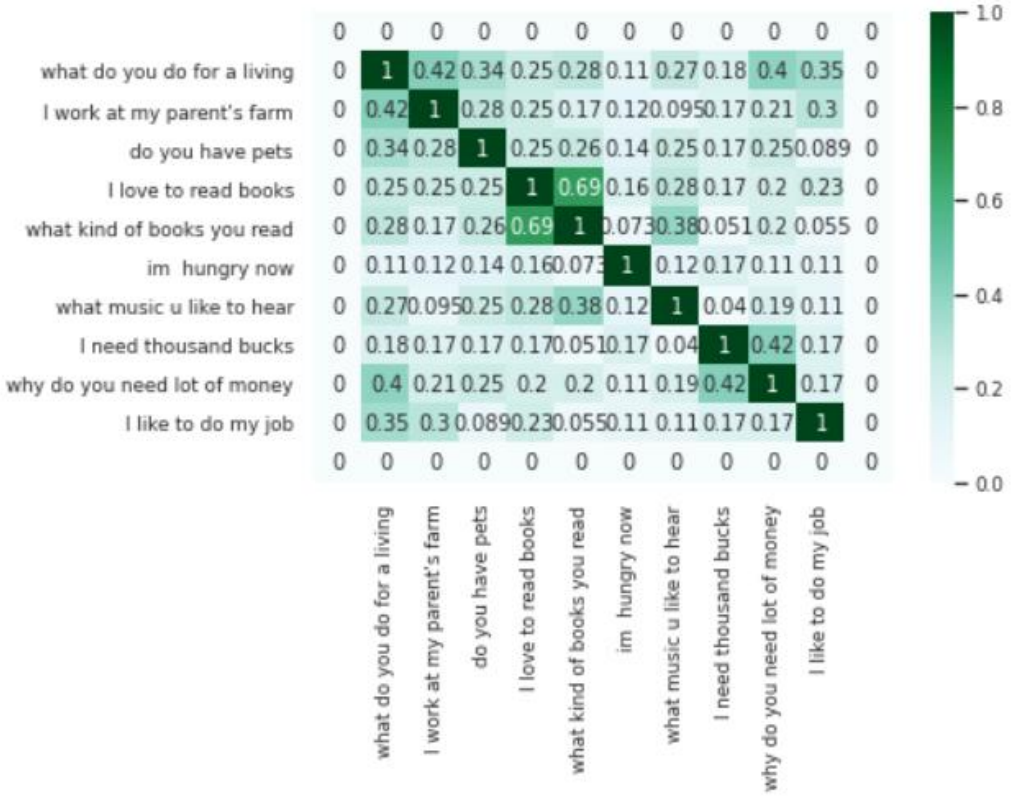

Figure 4.7 Sentence correlation or relevance matrix using ConveRT’s similarity score

After unit testing the quality of ConveRT’s embeddings, the sentences are passed through the declared function, the embedding 1024D vector of request and response are obtained. Distance vector is calculated and then passed through a sequence of

layers, Dense layer with 256 hidden units and dropout of 0.5 – Dense layer with 20 hidden units – Dense layer with 10 hidden units and dropout of 0.4 – Dense layer with 1 hidden unit of sigmoid activation. The total number of trainable parameters becomes 267,761.

Through the 20 epochs, the improvement of training and validation accuracy is witnessed as seen in Figure 4.8

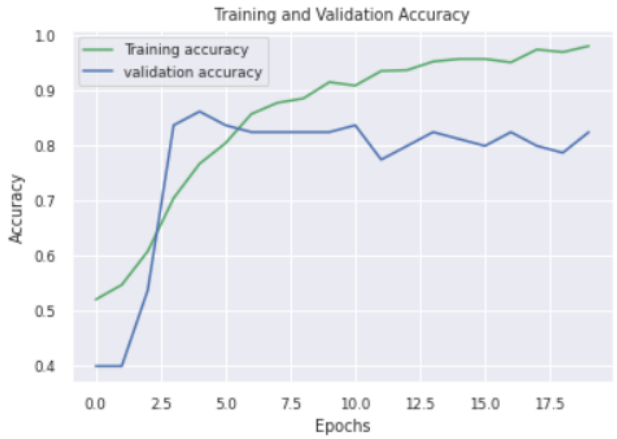

Figure 4.8 Training and validation Accuracy progress through epochs – Variant 4

It can be seen that the validation accuracy dropped at the 10th epoch and gradually improved then with ups ad downs. Even though the validation accuracy went greater than training accuracy on the second epoch, which could be due to random coincidence of the predicted label with the actual label because validation accuracy can never be greater than training accuracy, later as the model gets through further epochs, it learns the data nuances which can be witnessed as the validation accuracy grows gradually with ups and downs after the second epoch.

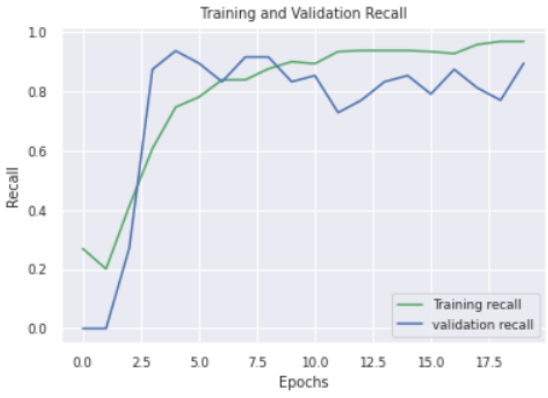

Figure 4.9 Training and Validation Recall progress through epochs – Variant 4

As seen in the above figure 4.9, the recall scores are better than accuracy scores but the pattern of growth looks the same.

A 10-fold cross validation is done which resulted in a list of 10 accuracy scores and recall scores as in Table 4.6

	List of 10 Scores	Average
Accuracy	[77.78, 88.89, 77.5, 61.25, 72.5, 75.0, 81.25, 67.5, 78.75, 83.75]	76.42
Recall	[76.32, 84.62, 78.72, 82.14, 75.86, 70.27, 80.49, 54.76, 72.22, 85.11]	76.05

Table 4.6 Accuracy and Recall from 10-fold cross validation – Variant 4

The 10-fold cross validation results of Recall and Accuracy are 76 and 73 respectively. which are obtained by comparing model predicted labels against the gold standard data labels look good enough inspite of the less volume dataset. Also, out of the 10 recall values, six are above 75 out of which two are around 85 which is outstanding in the so far experiments. This performance score of recall is obtained from the model which was trained with just 720 instances. Also, this performance resulted out of 10-fold cross validation, proving the generalisation of the model on the unseen data. In overall, as a result of first stage of evaluation this variant of ConveRT's pre-trained embeddings type is good at detecting confusion in the conversation, proving a human comparable performance.

4.6 State-of-the-art vs Variant 4

As witnessed earlier, A sentence similarity/correlation matrix shown the cosine similarity between the sentences which were embedded by ConveRT's embeddings. The same sentences were passed through this variant which uses pre-trained ConveRT's embeddings, process further by calculating distance vector, predict the probability score through some feed forward layers and final sigmoid layer. On an

interesting note, the results of this model were better than that of ConveRT's similarity score as shown in the below Table 4.7

Request	Response	ConveRT's score	Variant 4	
			score	label
I am on the soccer team	*I hate math class*	0.21	0.06441	Confusion
do you have pets	*I love baking cakes*	0.18	0.03569	Confusion
what kind of books you read	*I love to read comic books*	0.56	0.98212	No confusion
I am hungry now	*what music u like to hear*	0.38	0.01806	Confusion
what are you doing? I am listening to some classical	*Practicing for my SATs, I've got my test coming up this weekend*	0.29	0.907107	No confusion

Table 4.7 Similarity/Dissimilarity score – ConveRT vs Variant 4

The similarity scores of this variant is better than the original ConveRT's embeddings' similarity score. This is because of enhancing and finetuning the ConveRT's embeddings through combiner stage and further feed forward layers. The distance vector calculated at the combiner stage primarily aided this improvement. The laid architecture of the proposed design is so successful that it improved the state of the art performance of original ConveRT.

4.7 Second stage of evaluation – Across Variants

As per the second stage of our evaluation, which is Evaluation across the variants, the average recall scores of all the 4 variants are compared as seen in the below Table 4.8 variant 4 which is ConveRT pre-trained embeddings type has the highest value of 77 recall score. Also, it can be seen that there is a good difference between the scores of variant 4 and other variants. Hence variant 4 is declared to be the Best Performant of all the experimented variants to achieve the highest human comparable performance.

Variant #	Type	Accuracy	Recall
1	Self-Trained Embeddings	**58**	**57**
2	BERT Pre-trained Embeddings	**63.71**	**63.21**
3	GLOVE Pre-trained Embeddings	**73.04**	**68.95**
4	ConveRT Pre-trained Embeddings	**76.42**	**76.05**

Table 4.8 Second stage of evaluation results across variants

As ConveRT proves to be the state-of-the-art model in conversational response selection task, variant 4 which uses its pre-trained embeddings obtained the maximum recall score among all the variants. Reasons could be the architectural similarities between original ConveRT and variant 4 of ConveRT's embeddings type. Also, the native ConveRT is a domain specific model which trains the word embeddings on conversation corpus, whereas the word embeddings of BERT and GLOVE are trained on general word corpuses like Wikipedia. Though convert is actually trained on corpus size of 10M sentences which is lesser than BERT and GLOVE, the dual encoder-combiner-feedforward architecture of ConveRT made the edge more specific to the task of computing relationship between 2 objects such as a request and a response.

4.8 Third Stage of Evaluation – Best Performant vs Baseline

Variant 4, Pre-trained ConveRT embeddings type enters the third stage of evaluation of Best Performant vs Baseline. It is chosen to be compared against the baseline performance of human evaluation scores. Accuracy and recall scores of the human performance was calculated earlier just to check the overall performance of the human labelling using the entire data set labels. As stated earlier, a list of accuracy and k recall scores are computed for the human labelling when the best performant is decided. Hence, during the 10-fold cross validation of variant 4, at every iteration when the labels predicted by variant 4 on the test set are compared against the gold standard data labels, the human predicted labels of same test proportion are also compared against those self-marked gold standard data labels. so a list of 10 accuracy/recall scores is obtained for both variant 4 and Human performance. The below Table 4.9 lists the Human performance accuracy and recall scores compared to that of the best performant.

Performance	Baseline - human labelling	Best Performant - Variant 4
List of Accuracy scores	[86,88,90,92.5,91.25, 97.5,92.5,87.5,92.5,95]	[77.78, 88.89, 77.5, 61.25, 72.5, 75.0, 81.25, 67.5, 78.75, 83.75]
Average Accuracy	91.4	76.417
List of Recall scores	[82.97,83.78,85.29 ,87.5,86, 97,91,77,86,88.8]	[76.32, 84.62, 78.72, 82.14, 75.86, 70.27, 80.49 , 54.76, 72.22, 85.11]
Average Recall	86.77	76.05

Table 4.9 Third stage of evaluation: Baseline vs Best performant

The two lists of 10 recall scores of the variant 4 and baseline performance are subjected to Normality test before t-test. Because of small sample size of 10 values, it is recommended to used Shapiro-Wilk normality test. p value turned out to be 0.600 for the baseline performance distribution and for the variant 4, the p value turned out to be 0.080. Both the p values are greater than the alpha level of 0.05, hence null hypothesis is accepted for both the cases which means both the distributions are normal.

Accounting to normal distribution, and no dependency between these two isolated groups, Independent 2 sample t-test is to be performed at an alpha level of 0.05 and confidence of 95%.

t-statistic	p-value
3.298	0.003

Table 4.10 t-test results on hypothesis

The p-value is less than the alpha which means the null hypothesis that the two groups are similar is rejected. As in this case, the recall scores of base line performance and the proposed best performing model are not equal, there are good differences in the mean values of both the recall score distributions. Base line human performance scores are much greater than the proposed models' scores. Proposed models' performance in detecting the confusion of human participant in conversation with a conversational

agent is lesser than human's performance in spotting the same confusion. This proves that the performance of the proposed model is not nearly/approximately equal or comparable to human performance in detecting confusion and measuring the coherence of conversation. Statistically, t-test results convey that there is no good evidence to reject the null hypothesis of the research, ending up in accepting it, if the coherence of human conversation with a conversational agent is measured by means of semantic embeddings, it cannot achieve a human comparable performance. In statistical form, this result is reported as,

An independent-samples t-test was conducted to compare recall scores for human performance and the best performing variant (Variant 4). Significant difference in the scores for recall was found (M=76.05, SD= 8.49 for human performance, M= 86.48, SD= 4.23 for best performant), (t(10)= 3.298, p = 0.003)

Hence all the four variants are developed, trained and evaluated through all the three stages of evaluation and the framed hypothesis of the research is tested. The practical work along with all the data inputs in form of python notebooks and shared for the reference [2].

4.9 Discussion

The baseline performance which is evaluated as performance of external annotator labelling against the gold standard data self-marked labels, had an accuracy of 91% but the recall value was 87%. Though the consistency between the annotators was proved to be strong, The less recall of the second human's scoring shown that there is good space for research to prove a better ability of spotting confusion than humans by means of machine learning.

On evaluating individual variant against gold standard data self-trained embeddings type performed poor. The organic form of proposed design is actually the variant 1, having no influence of pre-trained embeddings. but it had a low performance as witnessed. Having 800 samples, around 10000 words, 1627 vocabulary size, achieving

[2] https://drive.google.com/drive/folders/1k5IDUGsfVW_hmkX2zCu04lLQ5Xx_2w0r

such a performance of 58% is reasonable. ConveRT is the state of art performer in response selection with a score of 72%, this number was obtained by training over a huge number of 10M sentences. Hence this performance cannot be treated as a poor performance but deliberately restricted due to the sample size.

Another important observation was the effectiveness of pre-trained embeddings which was positive. Starting with BERT, though both the accuracy and recall was 63%, It was better than the variant1 but way less than expected from the star performer of NLP. BERT has a complex architecture. BERT does not eject out just a n-dimensional vector for a sentence, It actually throws out three n-dimensional lists as Masks, Ids and Tokens. Fitting this output into a different architecture is resulting in a compatibility issue. Also, BERT is trained on general corpus, using it for conversation type classifications is another reason for the poor results. These results again prove the context of the previous work by Cer et al., (2018) which claims BERT's non-suitability for conversation type texts

Glove's 300 dimensional embeddings performed better than BERT type with accuracy and recall as 73 and 68 respectively. Actually, before experimenting Glove's 300 dimensional embeddings, it's 50 and 100 dimensional embeddings were attempted just to check which to be used in the variant. 300D type performed better than 100D type which performed better than 50D type. This observation made sure that higher the embedding size and higher the number of training samples, higher will the performance of embeddings. Glove is also easily incorporable into any architecture unlike BERT. Also glove works on co-occurrence and association of words similar to past research works that measured coherence using word association profiles. Hence it can be concluded that the concept of co-occurrence proved to be good in measuring coherence in this research also as it did in the past works by Beata et al., (2013) and Mohsen et al., (2016).

ConveRT embedding type variant performed the best among all the variants, having an accuracy and recall of 76.42 and 76.05 respectively. Also, the Euclidean distance example which had values of 18 for a relevant pair and 43 for a non-relevant pair, again proved the quality of ConveRT and the truthfulness of research's concept of Euclidean distance and embeddings association. This concept got enlightened when the variant 4's probability scores of all the undertaken examples were better than the native ConveRT's score by an average of 0.3 units on a scale of 0 to 1. Association of

similarity/dissimilarity metric such as Euclidean distance, influenced the embeddings to learn more from the distance vector which actually enhanced the original conveRT's state of art performance.

On comparing the conveRT's embedding type against the human performance, there were good differences as variant4 - human of 91-76 on average accuracy and 86-76 on average recall scores. Though on a deeper look on the list of accuracies and recall scores, 3 iterations out of 10-fold experiments of human and variant4 had almost equal values for accuracy and recall, 3 iterations had less differences and 4 iterations had good differences. If it was measured on hold-out validation there are higher chances that the variant 4 would have met human results. The four poor results pulled the t-test results against the alternate hypothesis. Considering the baseline performance, which is human scoring, it is difficult for any model to achieve it as humans are always better than machines on sensible classifications, variant 4 meeting human performance in 3 iterations is actually an accomplishment. Individually, Pre-trained embeddings had improved performance over self-trained embeddings, variant 4, ConveRT's embedding type individually had good performance against that of human scores while variant 3, GLOVE type, the performance was decent. Variant 1, self-trained embeddings type, on future works where it is trained on huge corpus, has the potential to achieve great results.

5. CONCLUSION

This chapter provides an overview of the research undertaken and the problem dealt with. A summary of the proposed design, experiments done, evaluation and results of those experiments is given. Also, the impact and the contribution that the research made is briefly discussed. A final section on leveraging the current work in possible directions in future is added.

5.1 Research Overview

Lot of human efforts have been replaced by machines and one such successful replacement is having a conversational agent to communicate in place of a human with another human. The basic and the important quality of these agents is to give relevant responses to human requests and maintain a coherent conversation with human. But they struggle a lot in giving such quality responses. A human tends to get confused when the agent gives an irrelevant or invalid response. Such a conversation becomes in-coherent. Looking for a proper means to measure the coherence of such human conversations with the agents, confusion was chosen as a parameter to measure the it in this research. When a confusion can be detected successfully in a conversation, it helps the agent avoid giving responses that incur confusion beforehand. Learning semantic embeddings through similarity/dissimilarity metrics such as Euclidean distance to detect such confusion was attempted.

5.2 Problem Definition

The research problem was that if there is a possibility to measure the coherence of human conversation with a conversational agent by means of semantic embeddings and achieve a performance which can be compared to that of human's ability to assess the coherence of a conversation.

A good conversation dataset ConvAI.io (Logacheva et al., 2018) was chosen to be used to measure the coherence. Based on the research problem, the null and the alternate hypothesis were defined. As the critical concept of the problem undertaken was semantic embeddings, several techniques and designs to generate semantic embeddings were experimented along with the experiments that use pre-trained

semantic embeddings. The model's performance was compared against each other and against the real human scores which formed the baseline performance.

5.3. Design/Experimentation, Evaluation & Results

The ConvAI.io dataset was analysed and 800 single turn clean human to bot conversations were picked and these conversations were self-labelled as 'confusion' or 'no confusion'. To prove the consistency of this self-labelling process, an external annotator was appointed to label the same conversations and Cohen-kappa score was calculated which turned to be 0.8, proving a perfect consistency, the self-labelled data became the gold standard data of the entire set of experiments. Data pre-processing steps of stop words removal, lemmatisation, vocabulary build, trimming, padding and data splitting were done to shape the data.

Base line performance was fixed as the performance of the external annotator compared against the gold standard data. A dual encoder neural network was designed with the components of embedding layer, encoder, combiner, feed forward layer. Four variants of this architecture were to be modelled with changes primarily in the embedding layer. Self-trained embeddings, BERT embeddings, GLOVE embeddings and ConveRT embeddings were experimented.

Recall was chosen as the primary performance metric used for the comparison of models, in order to capture as many confusions as the model could. Three stages of evaluation were done, and 10-fold cross validation was carried out at every stage. Stage 1, as every variant was individually tested against the gold standard data labels, resulted in variant 1 and 2 having poor scores, variant 3 having decent performance and varaint 4 having human comparable high performance. Stage 2, as to find the best performer among all the variants against the gold standard data, turned out to be variant 4 - ConveRT's embeddings type as the best performer. Stage 3 as to compare the best performer and the baseline human performance, the evaluation results were statistically tested by independent t-test with a confidence of 95%. The test indicated that the performance of the designed best performant is not comparable to real human performance. As a result, the alternate hypothesis was rejected, and the null hypothesis was accepted.

5.4. Contributions and impact

The technical purpose of this paper is to check if coherence of conversation or confusion in a conversation can be detected by means of semantic embeddings. So, a design was proposed to train semantic embeddings from the scratch with a dual encoder neural architecture comprising of a crucial stage called combiner. Training embeddings to maximise and minimise this vector based upon the confuion label is a different way to learn semantic embeddings. Even though this architecture is based on ConveRT (Henderson et al., 2019), some enhancements have been made to the original idea by means of calculating the distance between the vectors in the combiner and extending the ConveRT's network with few more feed forward layers under the intuition that the embeddings learn more from this distance vector. The improved results were seen when the Convert's embeddings were further trained in the new architecture. So far in the conversational domain, ConveRT's architecture has achieved the state of art performance in response selection by training semantic embeddings which were proved to be better than BERT in the conversational dataset. Since the architecture of this paper adds up to ConveRT's design and also to its performance which is so far the best in response selection process, A strong impact has been made by further enhancing the domain best model to generate improved semantic embeddings.

This research would mainly contribute to two spaces, coherence modelling specialised to confusion detection and neural architecture for training semantic embeddings in conversation domain. Future works can use architecture of the variant 1 (self-trained e,beddings type) to train semantic embeddings from scratch using any conversational corpus or use variant 4 (ConveRT's pre-trained embeddings type) to get enhanced ConveRT's embeddings and enjoy the top quality of embeddings in their downstream tasks. As discussed in literature works there are no solid researches that deal with confusion detection without using external expensive elements. This research has its biggest impact on confusion modelling which can be used to enhance quality of both human-machine and human-human communication in this digital world.

5.5. Future Work

The research has got some good scope for future work. Application wise, this architecture to decide on coherence can be extended to any objects and not only sentences or words. These objects are to be pairable in nature. Customer-product in the area of product recommendation, movie-genre in the area of genre and movie recommendation etc can be used as paired objects to be the inputs of the dual encoder and see how good they are related to each other. Architecture wise, the combiner could be enhanced with a hyper parameter to choose the type of merging like concatenation, subtraction, addition etc to check how the results vary with respect to every choice.

Extending to use the model for human to human conversation dataset could be a good immediate direction and enhancing the model for multi turn conversation could follow. Volume of the dataset could be increased, get a bigger mechanical turk to label the dataset and check the consistency of labelling across many annotators to create more trust-worthy bigger dataset. Bigger dataset would lead to better results on the self-trained embedding type. The size of the embedding layer could also be increased for the self-trained embedding type from 10 to learn a bigger efficient embedding space. The basic LSTM memory units of this architecture could be replaced by a multi head or self-attention, shared parameters architecture so as to enjoy the improvements from the original ConveRT's architecture and also to handle lengthy sentences. Number of feed forward layers could be increased after the combiner stage to improve the impact of merging on the learning of the semantics by the embeddings. Also, using this work as a service for downstream tasks, two modes could be setup, supervised and unsupervised mode that can be tweaked by the user where the supervised mode would give the similarity score or the category, unsupervised mode to train embeddings for paired objects.

BIBLIOGRAPHY

Alberto, T. C., Lochter, J. V., & Almeida, T. A. (2015, December). Tubespam: Comment spam filtering on youtube. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)* (pp. 138-143). IEEE. doi: 10.1109/ICMLA.2015.37

Banchs, R. E., & Li, H. (2012, July). IRIS: a chat-oriented dialogue system based on the vector space model. In *Proceedings of the ACL 2012 System Demonstrations* (pp. 37-42).

Bardenet, R., Brendel, M., Kégl, B., & Sebag, M. (2013, February). Collaborative hyperparameter tuning. In *International conference on machine learning* (pp. 199-207).

Barzilay, R., & Lapata, M. (2008). Modeling local coherence: An entity-based approach. *Computational Linguistics*, *34*(1),1-34. doi: 10.1162/coli.2008.34.1.1

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.". Brandtzaeg, P. B., & Følstad, A. (2017). Why people use chatbots., (pp. 377-392).

Casanueva, I., Temčinas, T., Gerz, D., Henderson, M., & Vulić, I. (2020). Efficient Intent Detection with Dual Sentence Encoders. *arXiv preprint arXiv:2003.04807*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*. doi: 10.3115/v1/D14-1179

Clark, L., Pantidi, N., Cooney, O., Doyle, P., Garaialde, D., Edwards, J., ... & Wade, V. (2019, May). What makes a good conversation? challenges in designing truly conversational agents. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1-12). doi: 10.1145/3290605.3300705

Cortes, C., Mohri, M., & Rostamizadeh, A. (2012). L2 regularization for learning kernels. *arXiv preprint arXiv:1205.2653*.

Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., John, R. S., ... & Sung, Y. H. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175*. doi: 10.18653/v1/D18-2029

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dong, F., Zhang, Y., & Yang, J. (2017, August). Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)* (pp. 153-162). doi: 10.18653/v1/K17-1017

Elsner, M., & Charniak, E. (2011, June). Extending the entity grid with entity-specific features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (pp. 125-129).

Feng, V. W., & Hirst, G. (2012, April). Extending the entity-based coherence model with multiple ranks. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 315-324).

Fu, R., Zhang, Z., & Li, L. (2016, November). Using LSTM and GRU neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)* (pp. 324-328). IEEE. doi: 10.1109/YAC.2016.7804912

Furlan, B., Batanović, V., & Nikolić, B. (2013). Semantic similarity of short texts in languages with a deficient natural language processing support. *Decision Support Systems*, *55*(3), 710-719. doi: 10.1016/j.dss.2013.02.002

Geller, S. A., Hoernle, N., Gal, K., Segal, A., Zhang, A. X., Karger, D., ... & Igo, M. (2020, March). # Confused and beyond: detecting confusion in course forums using students' hashtags. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge* (pp. 589-594). doi: 10.1145/3375462.3375485

Gharatkar, S., Ingle, A., Naik, T., & Save, A. (2017, March). Review preprocessing using data cleaning and stemming technique. In *2017 international conference on innovations in information, embedded and communication systems (iciiecs)* (pp. 1-4). IEEE. doi: 10.1109/ICIIECS.2017.8276011

Goldberg, Y., & Levy, O. (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Grosz, B. J., Joshi, A. K., & Weinstein, S. (1995). Centering: A framework for modelling the local coherence of discourse.

Guzman, A. L. (2018). What is human-machine communication, anyway. *Human-machine communication: Rethinking communication, technology, and ourselves*, 1-28.

Hancock, B., Bordes, A., Mazare, P. E., & Weston, J. (2019). Learning from dialogue after deployment: Feed yourself, chatbot!. *arXiv preprint arXiv:1901.05415*.

Hashimoto, C., & Sassano, M. (2018, April). Detecting absurd conversations from intelligent assistant logs by exploiting user feedback utterances. In *Proceedings of the 2018 World Wide Web Conference* (pp. 147-156). doi: 10.1145/3178876.3185992

Henderson, M., Casanueva, I., Mrkšić, N., Su, P. H., & Vulić, I. (2019). ConveRT: Efficient and accurate conversational representations from transformers. *arXiv preprint arXiv:1911.03688*.

Higashinaka, R., Mizukami, M., Funakoshi, K., Araki, M., Tsukahara, H., & Kobayashi, Y. (2015, September). Fatal or not? Finding errors that lead to dialogue breakdowns in chat-oriented dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 2243-2248). doi: 10.18653/v1/D15-1268

Hill, J., Ford, W. R., & Farreras, I. G. (2015). Real conversations with artificial intelligence: A comparison between human–human online conversations and human–chatbot conversations. *Computers in human behavior*, *49*, 245-250. doi: 10.1016/j.chb.2015.02.026

Islam, M. M., Sarkhel, S., & Venugopal, D. (2019, July). On Lifted Inference Using Neural Embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, pp. 7916-7923). doi: 10.1609/aaai.v33i01.33017916

Jiang, J., Hassan Awadallah, A., Jones, R., Ozertem, U., Zitouni, I., Gurunath Kulkarni, R., & Khan, O. Z. (2015, May). Automatic online evaluation of intelligent assistants. In *Proceedings of the 24th International Conference on World Wide Web* (pp. 506-516). doi: 0.1145/2736277.2741669

Jurafsky, D., & Martin, J. H. (2017). Dialog systems and chatbots. *Speech and language processing*, *3*.

Kiela, D., Hill, F., & Clark, S. (2015, September). Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural LanguageProcessing* (pp.2044-2048). doi: 10.18653/v1/D15-1242

Kim, T. K. (2015). T test as a parametric statistic. *Korean journal of anesthesiology*, *68*(6), 540. doi: 10.4097/kjae.2015.68.6.540

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*. doi: 10.3115/v1/D14-1181

Kiseleva, J., Williams, K., Hassan Awadallah, A., Crook, A. C., Zitouni, I., & Anastasakos, T. (2016, July). Predicting user satisfaction with intelligent assistants. In *Proceedings of the 39th International ACM SIGIR conference on*

Research and Development in Information Retrieval (pp. 45-54). doi: 10.1145/2911451.2911521

Klebanov, B. B., & Flor, M. (2013, August). Word association profiles and their use for automated scoring of essays. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1148-1158).

Krahmer, E., Swerts, M., Theune, M., & Weegels, M. (2001). Error detection in spoken human-machine interaction. *International journal of speech technology*, *4*(1), 19-30.

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015, February). Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Lévesque, J. C., Durand, A., Gagné, C., & Sabourin, R. (2017, May). Bayesian optimization for conditional hyperparameter spaces. In *2017 International Joint Conference on Neural Networks (IJCNN)* (pp. 286-293). IEEE. doi: 10.1109/IJCNN.2017.7965867

Li, J., & Hovy, E. (2014, October). A model of coherence based on distributed sentence representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 2039-2048).

Li, J., & Jurafsky, D. (2016). Neural net models for open-domain discourse coherence. *arXiv preprint arXiv:1606.01545*. doi: 10.18653/v1/D17-1019

Liao, Z., & Carneiro, G. (2016, March). On the importance of normalisation layers in deep learning with piecewise linear activation units. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 1-8). IEEE.

Logacheva, V., Burtsev, M., Malykh, V., Polulyakh, V., & Seliverstov, A. (2018). ConvAI Dataset of Topic-Oriented Human-to-Chatbot Dialogues. In *The NIPS'17 Competition: Building Intelligent Systems* (pp. 47-57). Springer, Cham.

Louis, A., & Nenkova, A. (2012, July). A coherence model based on syntactic patterns. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 1157-1168).

Meena, R., Lopes, J., Skantze, G., & Gustafson, J. (2015, September). Automatic detection of miscommunication in spoken dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (pp. 354-363).

Mesgar, M., & Strube, M. (2018). A neural local coherence model for text quality assessment. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 4328-4339).

Mesgar, M., & Strube, M. (2016, June). Lexical coherence graph modeling using word embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 1414-1423).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

Zhang, M., Feng, V. W., Qin, B., Hirst, G., Liu, T., & Huang, J. (2015). Encoding world knowledge in the evaluation of local coherence. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 1087-1096).

Nguyen, Dat Tien, and Shafiq Joty. "A neural local coherence model." *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017.

Niculescu, A. I., Yeo, K. H., D'Haro, L. F., Kim, S., Jiang, R., & Banchs, R. E. (2014, December). Design and evaluation of a conversational agent for the touristic domain. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific* (pp. 1-10). IEEE. doi: 10.1109/APSIPA.2014.7041744

Ortiz, C. L. (2014). The road to natural conversational speech interfaces. *IEEE Internet Computing*, *18*(2), 74-78. doi: 10.1109/MIC.2014.36

Pasupalak, S., Pantony, J. R., Hsu, W., Wu, Z., Tregenza, P., Suleman, K., ... & Ismail, T. (2017). *U.S. Patent No. 9,575,963*. Washington, DC: U.S. Patent and Trademark Office.

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

Pitsilis, G. K., Ramampiaro, H., & Langseth, H. (2018). Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

Rahman, A. M., Al Mamun, A., & Islam, A. (2017, December). Programming challenges of chatbot: Current and future prospective. In *2017 IEEE Region 10*

Humanitarian Technology Conference (R10-HTC) (pp. 75-78). IEEE. doi: 10.1109/R10-HTC.2017.8288910

Ramesh, K., Ravishankaran, S., Joshi, A., & Chandrasekaran, K. (2017, May). A survey of design techniques for conversational agents. In *International Conference on Information, Communication and Computing Technology* (pp. 336-350). Springer, Singapore. doi: 10.1007/978-3-030-15035-8_93

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (pp. 285-295).

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000, October). Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic commerce* (pp. 158-167).

Schmidt, M., Fung, G., & Rosales, R. (2007, September). Fast optimization methods for l1 regularization: A comparative study and two new approaches. In *European Conference on Machine Learning* (pp. 286-297). Springer, Berlin, Heidelberg.

See, A., Roller, S., Kiela, D., & Weston, J. (2019). What makes a good conversation? how controllable attributes affect human judgments. *arXiv preprint arXiv:1902.08654*.

Serban, I. V., Klinger, T., Tesauro, G., Talamadupula, K., Zhou, B., Bengio, Y., & Courville, A. (2016). Multiresolution recurrent neural networks: An application to dialogue response generation. *arXiv preprint arXiv:1606.00776*.

Sharma, Y., Agrawal, G., Jain, P., & Kumar, T. (2017, December). Vector representation of words for sentiment analysis using GloVe. In *2017 international conference on intelligent communication and computational techniques (icct)* (pp. 279-284). IEEE. doi: 10.1109/INTELCCT.2017.8324059.

Shearer, C. (2000). The CRISP-DM model: the new blueprint for data mining. *Journal of data warehousing*, *5*(4), 13-22.

Shum, H. Y., He, X. D., & Li, D. (2018). From Eliza to XiaoIce: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, *19*(1), 10-26. doi: 10.1631/FITEE.1700826

Somasundaran, S., Burstein, J., & Chodorow, M. (2014, August). Lexical chaining for measuring discourse coherence quality in test-taker essays. In *Proceedings of COLING 2014, the 25th International conference on computational linguistics: Technical papers* (pp. 950-961).

Srivastava, N. (2013). Improving neural networks with dropout. *University of Toronto*, *182*(566), 7.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104-3112).

Tan, X., Chen, J., He, D., Xia, Y., Qin, T., & Liu, T. Y. (2019). Multilingual neural machine translation with language clustering. *arXiv preprint arXiv:1908.09324*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

Wang, P., Xu, B., Xu, J., Tian, G., Liu, C. L., & Hao, H. (2016). Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing*, *174*, 806-814. doi: 10.1016/j.neucom.2015.09.096

Wang, X., & Yuan, C. (2016). Recent advances on human-computer dialogue. *CAAI Transactions on Intelligence Technology*, *1*(4), 303-312. doi: 10.1016/j.trit.2016.12.004

Webster, J. J., & Kit, C. (1992). Tokenization as the initial phase in NLP. In *COLING 1992 Volume 4: The 15th International Conference on Computational Linguistics*. doi: 10.3115/992424.992434

Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36-45. doi: 10.1145/365153.365168

Witten, I. H., & Frank, E. (2002). Data mining: practical machine learning tools and techniques with Java implementations. *Acm Sigmod Record*, *31*(1), 76-77.

Yadav, S., & Shukla, S. (2016, February). Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In *2016 IEEE 6th International conference on advanced computing (IACC)* (pp. 78-83). IEEE. doi**:** 10.1109/IACC.2016.25

Yang, Y., & Liu, X. (1999, August). A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 42-49). doi: 10.1145/312624.312647

Zeroual, I., & Lakhouaja, A. (2017, April). Arabic information retrieval: Stemming or lemmatization?. In *2017 Intelligent Systems and Computer Vision (ISCV)* (pp. 1-6). IEEE. doi**:** 10.1109/ISACV.2017.8054932

Zhou, X., Dong, D., Wu, H., Zhao, S., Yu, D., Tian, H., ... & Yan, R. (2016, November). Multi-view response selection for human-computer conversation.

In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 372-381).