Conversational agent with common-sense: Responding to nonsensical statements

by

Anandh Perumal Konar

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Conversational agents, also known as chatbots, are designed to have a real-time conversation with humans. Closed domain chatbots are limited to a specific task they're designed to do. They can be rule-based or information retrieval based chatbots while open domain chatbots are meant to mimic humans in conversation.

For a conversational agent to have a human-like conversation, the agent needs to generate grammatically correct and coherent output relative to the input. However, it is also crucial for the agent to have common sense and be equipped with basic facts about the world. For example,consider the following statement: "He gave birth to a baby." For humans, it is common sense that males cannot give birth. Therefore, the given statement is nonsensical. However, for a system, it is challenging to distinguish between sensical and nonsensical statements. Moreover, it is even more challenging for a system to generate an explanation stating why the given statement is nonsensical. We propose UNION, a unified end-to-end framework, to generate a meaningful explanation to a given nonsensical statement that utilizes several existing commonsense datasets to allows a model to learn more dynamically under the scope of commonsense reasoning. To perform model selection efficiently, accurately and promptly, we also propose a couple of auxiliary automatic evaluation metrics so that we can extensively compare the models from different perspectives. Our submitted system results in an excellent performance in the proposed metrics. It outperforms its competitors with the highest achieved score of 2.10 for human evaluation while maintaining a BLEU score of 15.7.

Preface

As part of this manuscript, our paper at SemEval 2020 was accepted but yet to be published.

1. Anandh Perumal, Chenyang Huang, Amine Trabelsi, Osmar R. Zaiane, "ANA at SemEval-2020 Task 4:mUlti-task learNIng for cOmmonsense reasoNing (UNION)", the 14th International Workshop on Semantic Evaluation collocated with COLING. [37]

The above paper is covered in Chapter 3. I contributed most of the ideas and experiments and writing. However, Dr. Osmar Zaiane, Dr. Trabelsi, and Chenyang also helped in constructive input. We are currently trying to publish our state-of-the-arts results in the CoS-E dataset in Section 4.3 in the form of conference paper or journal.

We are shaped by our thoughts; we become what we think.

- Gautam Buddha, Philosopher.

Acknowledgements

I want to begin by expressing my sincere thanks to my supervisor Dr. Osmar Zaiane. Without his support, I would have never been able to fulfill my thesis goals. Moreover, I would like to thank Chenyang Huang and Dr. Amine Trabelsi for helping me with my thesis and for the valuable discussions.

Last but not least, thanks to my family and friends for their continuous support and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Conversational Agents (CA), also known as chatbots, are designed to have real-time conversations with humans. Conversational agents fall into two categories Open Domain chatbots and Goal-oriented chatbots.

An Open Domain chatbot can have conversations with humans about a vast array of topics; it has a wide range of knowledge and mimics human-like conversation by keeping the user engaged.

A goal-oriented chatbot, on the other hand, has a predefined set of objectives. It helps the user accomplish a particular set of tasks and has minimal knowledge about domains outside of it is responsible tasks; it cannot answer any general question that lies outside its domain of expertise. For example, a closed domain chatbot for a restaurant can handle inquiries related to orders, their delivery, etc., but is incapable of answering questions related to weather, as that is outside its domain [18] [59]. Unlike open domain chatbots that can have more exciting and engaging conversations with users, the goal oriented chatbots meant o have short, task-oriented conversations.

Both open domain and goal-oriented chatbot respond to a specific question either by rule-based approach or by response generation.

The rule-based chatbots have a predefined heuristic approach for picking the best possible response for the given inputs and conversational history. This heuristic approach can be a simple rule-based template match or information retrieval or can depend on Machine Learning classifiers. Such systems do not generate a spontaneous response. Hence the response is more repetitive and leads to less interaction with the system.

Generative models, on the other hand, are typically based on deep learning techniques. They generate an impromptu response for each input by learning to predict over the sequence of words in the training data, resulting in the generative model producing a more diverse range of responses, similar to human conversations.

The biggest challenge encountered by chatbots in replicating a human-like conversation is the lack of common sense - the necessary information about the environment and the general awareness about the world. Another impediment is the need to generate responses that are coherent and sensible for any given input. Most chatbots generate responses on the assumption that given input is a sensical statement or question. When the given input is nonsensical, then the system fails to generate a sensical explanation of why the given input is nonsensical.

Nonsensical statement classification and explanation generation can help dementia detection. Dementia causes aphasia [10] [30], which is a comprehension and communication disorder. Aphasia is the loss of the ability to express oneself and understand, which may lead to the formation of a nonsensical statement. Thus, our proposed commonsense pipeline system can help detect the given nonsensical statements and help the user explain why the given statement is nonsensical. While detecting confabulation or nonsensical statements is of obvious value toward detecting hallucination or dementia, we do not prove in our work our position regarding the usefulness of explaining nonsensical statements for dementia disclosure. We leave the exploitation of this supposition to future work.

For a system to respond to a nonsensical statement, the system needs to have some common sense to ascertain why a given statement does not make sense and generate a response that is coherent to the given input. In this research, we study the challenges related to the classification of a sensical and nonsensical statement. If the given statement is nonsensical, then our model generates an explanation of why the given statement does not make sense.

1.2 Problem Statement

For chatbots to have conversations like humans, they need to understand the world and make judgments based on their knowledge. In humans, common sense may vary from person to person depending on various factors such as education, culture, religion, and other demographics.

C.S. Lewis [26] defined common sense as *sound practical judgment concerning everyday matters, or an essential ability to perceive, understand, and judge that is shared by ("common to") nearly all people.* Humans generally learn things in a structured way; for example, we learn basic maths before learning to apply the BEDMAS rule, and this better facilitates our understanding of algebra, in this case. This structured learning is applicable across various fields. Whether it is day-to-day conversations or sophisticated Quantum Physics, we need to have a basic understanding of topics to learn and progress. Thus knowledge is used to answer questions and make practical judgments.

Similarly, for a chatbot, to start being more lifelike, it needs to infer the knowledge that it has gained and make judgments based on that and generate a response accordingly.

If someone says "*the sky is green*" we know the sky is blue, the response should be "the sky is blue, not green." This example is common-sense reasoning since it requires some amount of general knowledge to answer such questions.

A rule-based chatbot cannot be used for common sense reasoning because, as discussed earlier, the rule-based approach will always be repetitive, and we cannot foresee all cases and define rules for all conditions. Another essential problem is that it cannot automatically generate a sensical response, which is imperative for common sense response generation. The response relies on the understanding of several statements or facts; some answers may need in-depth knowledge and the ability to co-relate between the things learned like human beings. For example, given "*Alberta is in Europe.*" as an input to a chatbot, then it needs to make a judgment based on its prior learning that Alberta is in Canada and Canada is in North America. It should generate a response, "Alberta is in North America." To develop such systems, information retrieval via a database or a rule-based system is not an easy task and gives rise to several questions.

1. What is considered common-sense in terms of a chatbot?

Unlike humans, expert systems, or Machine Learning chatbots, do not learn from the conversations they have with users. Their knowledge is limited to data on which they are initially trained. Therefore, we define common sense for the chatbot as not a domain, religion, or culture-specific, but rather based on general open-domain knowledge that includes common and acceptable facts across various groups of people, such as "Earth is covered with 3/4 water".

2. How to classify whether a given statement is sensical or not?

We train a discriminative model (Statement Discriminator) to classify whether the given statement is sensical or not. If the given statement is non-sensical, then further actions are taken.

3. How can a chatbot learn common sense and generate a response based on a nonsensical statement?

For reasoning, we train a generative model, mUlti-task learNIng for cOmmon-sense reasoNing (UNION), on open-domain knowledge that can generate an explanation to a given nonsensical statement.

1.3 Thesis Statement

In this study, we address the problem of common-sense reasoning in open domain chatbots: Deduct whether the given statement is against common sense and generates a valid explanation if the given statement is against common sense. The following is the deduction based on the research:

We hypothesis that *a language model can generate a common-sense response to a nonsensical statement provided the system has already common knowledge about the world in the form of a text corpus from which to learn the language model.*

1.4 Thesis Contribution

In this dissertation, we work mainly on common-sense reasoning for a nonsensical statement. The main contributions of this research are as follows:

1. Reward Augmented Model: We explore Reward Augmented Maximum Likelihood (RAML) for explanation generation. Our results show that RAML performs better than our baseline model.

2. UNION Model:
   We train and evaluate a generative model for common-sense reasoning. Our proposed model achieves state-of-the-art for common sense explanation generation. We also propose several automatic auxiliary metrics to evaluate common-sense NLU systems.

3. Common-sense system:
   We propose a pipeline for chatbots to inherit our common-sense system with minimal changes. We also train a discriminator model, Estimated Approval, which evaluates the generated reasoning for the nonsensical statement, making it easier to evaluate the generated explanation for any given input.

4. Evaluation metrics:
   The human evaluation for the generated explanation is expensive, and BLEU is not an appropriate measure as it does not consider the semantic similarity in the statements. Hence, we propose several automatic auxiliary metrics to evaluate the explanations generated by different models.

1.5 Thesis Organization

The remainder of the dissertation contains four chapters. In chapter 2, we discuss the various advanced deep learning techniques used in natural language processing. In chapter three, we focus on related work and our proposed architecture, and in chapter four, we will discuss the results of each model discussed in chapter 3. Chapter five address the contribution of this research and possible future work to extend this research. Having the above overview in mind, the four chapters in this research are as follows:

Chapter 2: Background and Related Work

We discuss in detail the advantage and disadvantages of using deep learning techniques for language models over statistical-based language models. We look into the latest deep-learning techniques for developing a generative language model such as Recurrent Neural Network, LSTM, transformers, and the earlier framework in common-sense reasoning and drawback of those systems.

Later, we describe the baseline models and the drawback of those models. Further, we will look into our proposed UNION architecture.

Chapter 3: Experiments and Results

We provide a detailed description of the different data sets we used in this research. Moreover, we will also discuss in detail the drawbacks of different metrics and advantage of our proposed metric for evaluation and also analyze the performance of the baseline model and our proposed model.

Chapter 4: Conclusion and Future work

In this chapter, we summarize the results and our contribution to the common-sense explanation generation system, and we look into possible future ways to extend this research.

Chapter 2

Background and Related work

Machine common sense has been one of the most challenging Artifical Intelligence (AI) problems, and a major, critical missing component of AI. It might be helpful in various aspects of day-to-day life. Below are the few of the broad uses given in [17] where common sense for AI could be helpful:

1. Sensemaking:
   Given any data or problem to the system, it needs to interpret and understand the various aspects of that data or problem based on real-world situations.

2. Reasoning Capabilities:
   A machine's common-sense reasoning can help monitor and understand AI systems while making a crucial decision.

3. Communication:
   We tend to assume while communicating with fellow humans that they have background knowledge about the world. For a machine to have a successful interaction with humans, it also needs to have basic common sense, which would help create more useful and exciting communication.

4. Transfer Learning:
   The common-sense knowledge of one domain could be helpful in other

areas too. For example, quantitative reasoning could be beneficial in chatbots as well as in AI systems for physics.

For humans, common-sense reasoning comes naturally and that helps us understand statements, but, it is a sophisticated process.

In this chapter, we explain the latest deep learning techniques for Natural Language Processing (NLP), that are mainly used for common-sense reasoning frame-works. Later, we discuss earlier work in this area and several fundamental concepts and critical issues for common-sense reasoning for machines. Finally, we introduce our proposed models, the UNION model and the baseline model.

2.1 Technical Background

2.1.1 Language Model

The language model is useful for several natural language processing applications such as information retrieval [28], machine translation [55] [50] and part-of-speech tagging [43]. A statistical language model estimates the probability distribution over the sequence of tokens in the training corpus. Given a sentence of length l, it estimates the probability for the whole sequence by computing the joint probability of the tokens in the given sentence. Thus, the sentence which appears the most in the corpus has a high probability compared to the sentence which appears less often. Therefore, any given sentence that is not grammatically correct will have a low likelihood, since such a sentence has not been encountered in the corpus.

$$P(w_1, ..., w_l) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_l|w_1, ..., w_{l-1}) \tag{2.1}$$

Equation 2.1, is the joint probability for the sentence of length l and $P(w_l|w_1, ..., w_{l-1})$ is the conditional probability for predicting the word w_5 that occurs in the sequence after $(w_1, ..., w_{l-1})$. However, computing the joint probability over the entire sequence has two significant limitations:

1. As the number of tokens in the training corpus increases, the number of possible combinations increases exponentially, and it is not feasible to have enough data to have all the possible combinations of tokens in a given language.
2. The computation costs increase as the sequence length increases.
3. How to prevent the language model from assigning zero probability to a sequence that did not occur in the training corpus.

Markov Assumption helps overcome the above problems by approximating the conditional probability, i.e., for each state, it only depends upon a few past states. Instead of computing joint probability over the entire sequence for predicting the next word, we solely depend on the past n words, which is called an $n - gram$ language model.

$$P(w_l|w_1, ..., w_{l-1}) \approx P(w_l|w_{l-1}) \tag{2.2}$$

Equation 2.2 estimates the probability of w_l depending only on the previous word w_{l-1}. This model is called a $bi - gram$ model. The most commonly used $n - gram$ models are unigram, bi-gram, and tri-gram. In the unigram model, the probability of each word in the given sentence depends on itself. Such models are used mainly in information retrieval tasks. The bigram depends upon the previous word, while the trigram depends on the previous two words. The n in the $n - gram$ model indicates the number of previous words on which the language model depends, on predicting the next word in the given sequence.

Smoothing techniques like Laplace Smoothing, Backoff, Interpolation, Kneser Ney, Additive Smoothing, etc., prevent assigning a zero probability to the sentence that did not occur over the training data. The $n - gram$ language model is parameter-less optimization that makes it easier to implement and train. However, the higher the n value the better the model is, which means much computation. Furthermore, it leads to a sparse representation of the language. Moreover, for conversational NLU systems, the statistical language model cannot spontaneously generate responses for the given input, making the conversation boring.

2.1.2 Deep Learning

Deep learning for language models has been very successful, as they learn complex representation data. Deep learning for NLP has the following advantages over the traditional statistical language model:

1. It reduces the efforts needed in developing handcrafted features

2. It learns multi-level abstraction, and is not limited NLP. It is applicable across various domains, and is one of the critical features of deep learning.

3. Recurrent Neural Networks (RNN) help capture the linguistic recursion in the language. Every language follows a specific set of rules and patterns such as grammatical structure and other linguistic elements. These patterns or rules repeat over the sequences, and these repeated regulations or patterns are known as linguistic recursion.

2.1.3 Word Embedding

Machine learning models only understand numeric data. Word embedding helps the text data convert into a numeric format. In Bag Of Words (BOW) embedding, each token in the vocabulary will have a unique one-hot vector representation, which makes the BOW a sparse representation as the number of the words in the vocabulary increases.

The word embedding learned through unsupervised deep learning techniques such as GloVe and Word2Vec has achieved great success in several NLP tasks.

Word2Vec and Global Vectors for Word Representation (GloVe) are two of the most successful and popular word embedding models. They're a continuous and dense vector representation for the token in the vocabulary. Moreover, they also capture the syntactic and semantic relationships between words.

Word2Vec has two popular models to convert word to vector, Continuous Bag-of-Words (CBOW) and Skip-gram, which was proposed by [32]. In CBOW, the training strategy is to predict the word given its context; Skip-gram to predicts the context given the word GloVe focuses on the co-occurrence of the words in the provided window of the sequence. The word embedding generated by these models does not consider the context of the sentence. The word embedding for "bank" in "John went to the river bank today", and "John visited the bank to withdraw his salary" has the same vector representation. ELMo [38] and Byte-pair [46] embedding overcome the contextual embedding problem in earlier embedding models. ELMo is a character-level language model that helps handle the out-of-vocabulary words. It uses a deep bidirectional LSTM architecture, which allows ELMo to learn the deep contextual meaning of each word. Byte-pair is a sub-word embedding, where rare words are broken down into sub-words to handle out-of-word vocabulary. Therefore, the word vector generated for the word "river" by Byte-pair and ELMo is different based on the context of the sentence.

2.1.4 Neural Language Model

Bengio [5] proposed the first Neural Language Model (NLM) which gave rise to several other neural language processing tasks using deep learning. An NLM model takes a fixed length of tokens as input to predict the next word in the sequence. The respective word embedding replaces each token in the sequence. Embeddings for each token are concatenated together as a matrix and passed as an input to the NLM, and later, to the feed-forward layers followed by a softmax layer.

$$z_t = f(\mathbf{W}^\intercal h_t + b) \tag{2.3}$$

In Equation 2.3 above, f is a non-linear activation function such as relu, tanh, sigmoid, W $\in \mathbb{R}^{\mathbb{V}\times\mathbb{K}}$ and b $\in \mathbb{R}^{\mathbb{V}}$, where V is the vocabulary size, and K is the dimension of the word embedding.

The output from the h_t is passed to the softmax activation function to

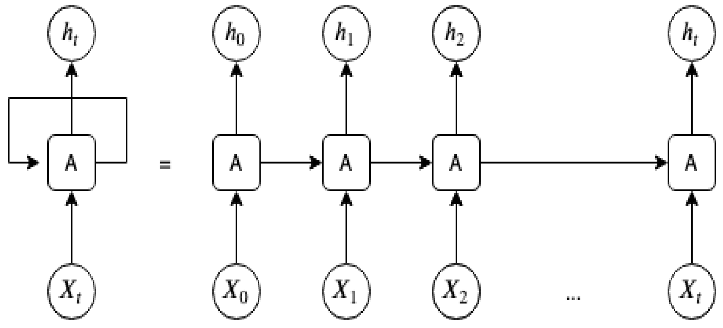

Figure 2.1: RNN encoder-decoder with attention mechanism

generate a conditional probability distribution to predict the next word in the sequence.

$$P(w_l|w_{l-n},...,w_{l-1}) = \frac{exp(z_t)}{\sum_{j=1}^{n} exp(z)} \tag{2.4}$$

To capture long term dependency in the sequence using a feed-forward neural network, the window size n needs to be big enough, which makes it practically impossible due to the computational requirements [33].

2.1.5 Recurrent Neural Network

Recurrent Neural Networks (RNN) [16] belongs to the family of Neural Networks, where output for each time step depends upon the current input and output from the previous time step as shown in the Figure 2.1 . RNN's have been very useful for several NLP tasks as they conditions on all the previous tokens generated in the past, which helps learn the long-term dependencies.

$$h_t = f(b + Wh_{t-1}Ux_t) \tag{2.5}$$

In Equation 2.5, h_t is the hidden state at timestep t. At every timestep, it takes the input from the previous timestep h_{t-1} and the input at that timestep x_t where W and U are the learnable parameters and f is the activation function. As at each timestep, the RNN depends upon the previous hidden cell state, which helps in learning long-term dependencies in the sequence. RNN's are uni-directional, i.e., depend only on past tokens. Thus, it makes it difficult to learn the given input representation.

The bidirectional RNN's overcome the above problem by having two-time directional RNNs, i.e., bidirectional. The forward state RNN learns positive-directional dependency while the backward state RNN learns the negative-time directional dependency. At each timestep, the information from both the RNN states is combined to generate the output for the respective timestep.

However, Pascanu shows in [36] that during backpropagation, the gradient of the loss function decays exponentially with time, which makes RNNs challenging to train in longer sequences despite their theoretical capability.

2.1.6 LSTM

The Long Short-Term Memory (LSTM) is a special kind of RNN that was introduced by Hochreiter and Schmidhuber in [19]. LSTM overcomes the vanishing gradient problem in RNNs with the help of three gates: an input gate, an output gate, and a forget gate. In Equation 2.6, the forget gate helps in removing the information that is no longer needed. In contrast, the input gate in Equation 2.7 adds new information to the cell state, and the information from the output gate in Equation 2.8 becomes the previous hidden state for the next timestep. The LSTM cell block is shown in Figure 2.2.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.6}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.7}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.8}$$

The key component on the LSTM is the cell state, denoted by C_t for the timestep t in the Equation 2.10. The cell state stores the relevant information in itself through the input gate and removes the irrelevant information with the help of the forget gate.

$$\hat{C} = tanh(W_c c_t + U_c h_{t-1} + b_c) \tag{2.9}$$

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{2.10}$$

$$h_t = o_t * \sigma(C_t) \tag{2.11}$$

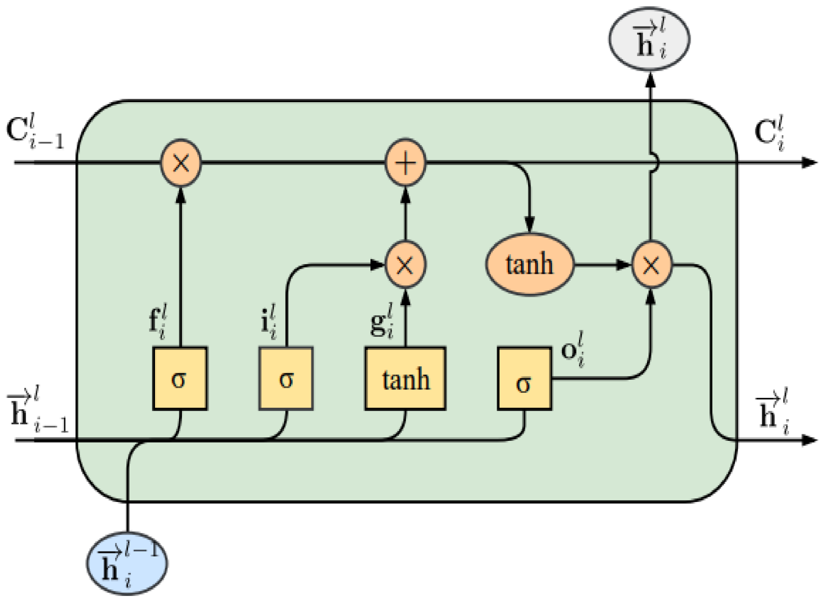

Figure 2.2: LSTM block `https://bit.ly/2zfn6y4`

LSTM architecture helps RNN to learn longterm dependencies and overcomes the traditional problem in the RNN, i.e., vanishing gradient. However, there are certain other limitations in the LSTM based language models. In the LSTM or RNN-LSTM models, the decoder relays on the single encoder vector to decode the complete information during decoding, ignoring essential or relevant information needed for the particular timestep during decoding.

2.1.7 Encoder - Decoder

The sequence-to-sequence (Seq2Seq) architecture in the Figure 2.3 proposed by Sutskever in [50] is the backbone for many natural language applications. The Seq2Seq model maps the given input sequence to the output sequence. The key architecture of Seq2Seq is an RNN or Transformers that helps to model on map varying length of input to output sequences. In the Section 2.1.9, we will discuss Seq2Seq architecture in detail using Transformers. The Seq2Seq architecture has two main components, the encoder, and the decoder, as shown in Figure 2.3.

The encoder is a stack of several RNNs, in each timestep, the RNN accepts

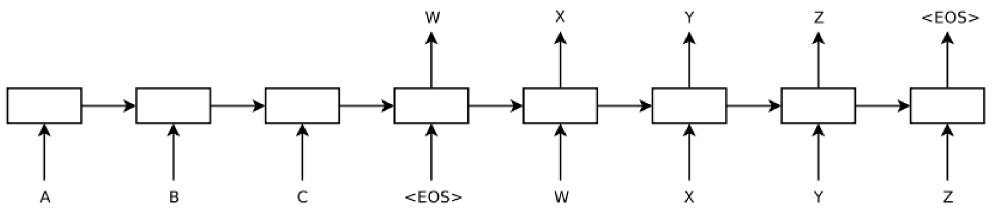

Figure 2.3: Encoder-Decoder Architecture. The "ABC" is the source statement to the model and "WXYZ" is the target sequence. The $< EOS >$ indicates the last token in the sequence. [57]

a token from the input sentence $X = x_1, x_2, ..., x_m$.

$$h_t = f(W^{hh}h_{t-1} + W^{hx}x_t) \tag{2.12}$$

Each token in the input sequence is translated to a vector of real numbers using word embeddings discussed in Section 2.1.3. The Equation 2.12 is used to compute the hidden states in the RNN, where t in the equation represents the timestep. The W^{hh} and W^{hx} is a learnable parameter and h_t is the hidden state at the timestep t. The final hidden vector, z, by the encoder, encapsulates the entire information in the input elements.

The decoder is another RNN model. During training, the target statement is augmented by adding $< EOS >$ as the last token to the statements. The hidden state of the decoder RNN is initialized with z.

The z helps the decoder to represent the complete information in the input sequence. A fully connected layer follows the RNN component. At each time step in the decoder, the RNN predicts a token $\hat{y}_t$ in the target sequence $\hat{Y}$.

$$\hat{y}_t = argmax(softmax(W^s h_t)) \tag{2.13}$$

In Equation 2.13, the $softmax$ function is used to compute a probability distribution over the target vocabulary. The $argmax$ function is used to select the next token $\hat{y}_t$ in the sequence with the highest probability. The predicted $\hat{y}_t$ is the input to the decoder in the following steps. When the decoder predicts the $< eos >$ with the highest probability, it indicates the last token in the

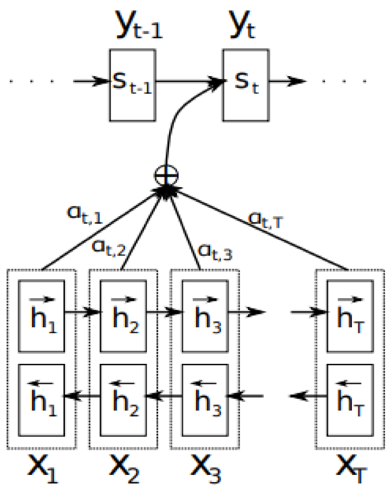

Figure 2.4: RNN Encoder-Decoder with attention mechanism [3]

sequence.

2.1.8 Attention Mechanism

The attention mechanism proposed by Bahdanau [3] solves the problem of depending on only the final vector from the encoder to decode the complete information. During the decoding step, the attention layer in between the encoder and the decoder generates a context vector. The context vector in Equation 2.14 helps the decoder pay more attention to important or relevant tokens and less attention to the irrelevant tokens at that timestep. The architecture of the RNN with attention mechanism shown in Figure 2.4.

$$c_i = \sum_{n=1}^{j} \alpha_{ij} h_j \tag{2.14}$$

The attention weight α_{ij} is a learnable parameter. The input to the attention layer in Equation 2.19 is the current target hidden state and the source hidden state.

$$\alpha_{ij} = \frac{exp(score(s_{t-1}, h_j)}{\sum_{k=1}^{N} exp(score(s_{t-1}, h_j)} \tag{2.15}$$

$$score(s_t, h_j) = \mathbf{v}^\intercal tanh(W_a[s_t; h_j]) \tag{2.16}$$

$\mathbf{v}^\intercal$ and W_a are learnable weight matrix.

2.1.9 Transformer

The Transformer architecture as shown in Figure 2.5 proposed by Vaswani [57] is a relatively new model compared to RNN. RNN consumes the input in every timestep; it helps the RNN incorporate the position of the word in the sequence. Transformers use pure attention mechanisms, and the whole sequence is passed as an input to the model. Thus, it fails to differentiate the position of the words in the sequence. The architecture of the Transformer model is shown in Figure 2.5.

Positional Encoding

Positional Encoding (PE) helps the transformers learn the position of each word in the sequence. PE is a time series embedding value that is added to each token in the input sequence $X = (x_0, x_1, ..., x_{L-1})$ to differentiate the position of each word in the sequence.

$$PE(pos, 2i) = sin(pos/10000^{2i/d_{model}}), \tag{2.17}$$

$$PE(pos, 2i+1) = cos(pos/10000^{2i/d_{model}}) \tag{2.18}$$

The cyclic nature of the $sin(x)$ and $cossin(x)$ functions in Equation 2.17 and 2.18, respectively, help to retain the positional information of the word in the sentence.

Transformer Architectures

The encoder of the transformer has an N identical layer; each of the layers consists of mainly multi-head, self-attention, and feed-forward networks, along with residual connections followed by a layer normalization after each part. The decoder of the transformer has masked multihead attention along

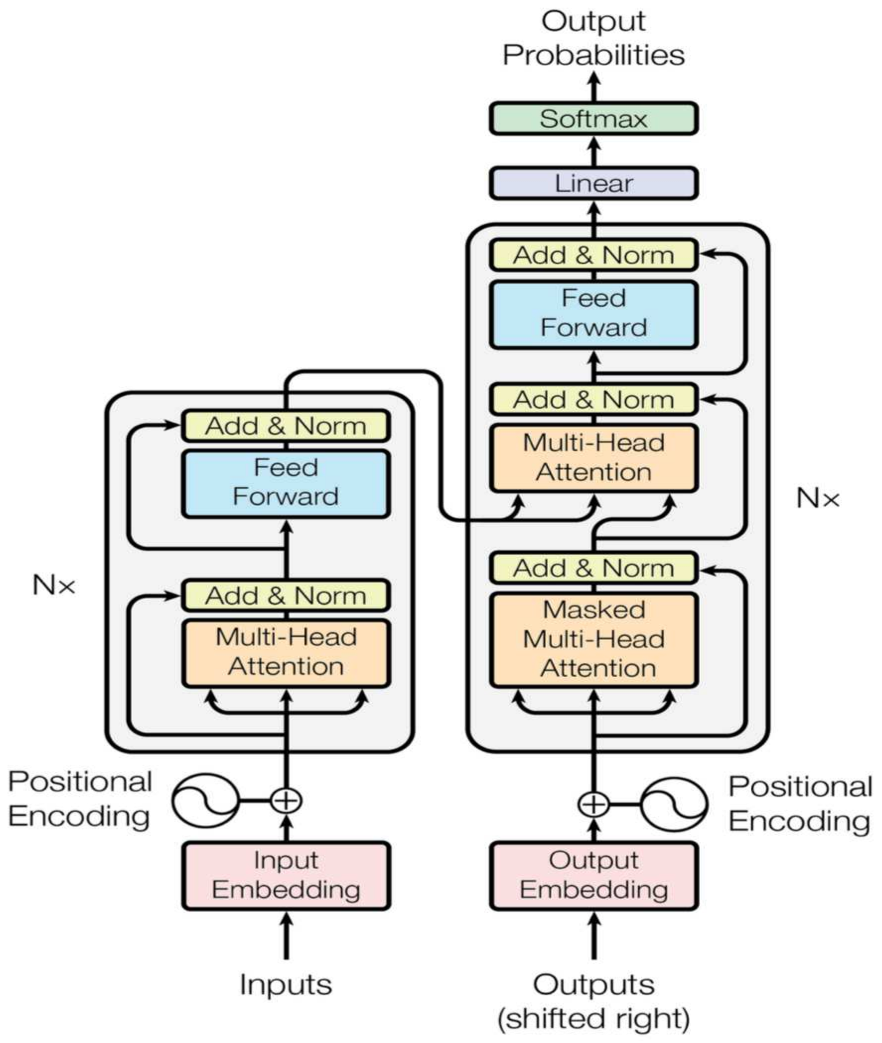

Figure 2.5: Transformers Encoder-Decoder Architecture [57]

with multi-head attention and feed-forward networks. The masked multi-head attention helps to prevent the flow of information from right-to-left, making the decoder behave in an autoregressive fashion. At each timestep, the decoder can only have access to the token it predicted so far, whereas, in the encoder, it has access to past and future tokens. The encoder and decoder in the transformer architecture have the same number of layers and attention-heads.

Each head in the self-attention layer has three major components, Queries (Q), Keys (K), and Value (V) all the three are learnable parameters. Equation 2.19 is scaled dot-product attention that is used to compute the attention matrix, where Q and K help in determining the importance of other words in the sequence while encoding the particular word. The weight matrix computed by $QK^\intercal$ is normalized to one where d_k is the dimension of both K and Q.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.19}$$

The multi-head in each layer helps to learn a more complex and different representation of the sentence, and each head has a varying range of contribution towards the final output.

$$MultiheadAttention(Q, K, V) = concat(head_1, ..., head_h)W^o \tag{2.20}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.21}$$

The attention weights in the encoder and decoder head do not share the parameters. The W^Q, W^K, and W^V are learnable parameters for each head, and W^O is the weight matrix for the feed-forward layer.

2.1.10 Multitask Learning

Caruana proposed the Multitask Learning (MTL) architecture for Neural Network in [8]. During MTL, the training signals from different related tasks help the model to better generalize and prevent overfitting.

The traditional Seq2Seq model usually solves a single problem, for example, sequence prediction, classification, structure prediction, where a single output is predicted at the end as shown in the Figure 2.6. Training multiple-tasks

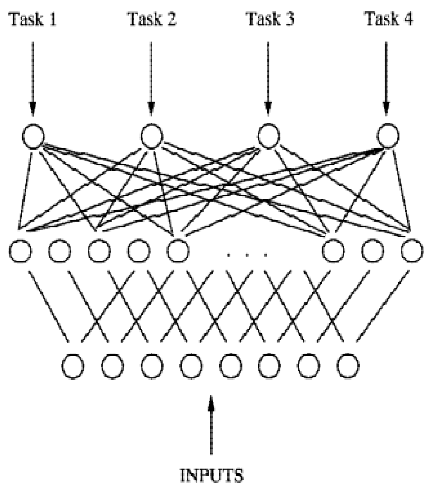

Figure 2.6: Multitask Learning of four tasks with the same inputs[8]

increases the noise in the data, and adding noise to backpropagation may help the model to generalize better [20]. Thus, MTL helps to reduce the overfitting [4] as the model has to simultaneously capture the representation of all the tasks. MTL is not limited to NLP, but is also applicable to various other domains, such as computer vision.

For MTL to work, all the subtasks must be related. The tasks need to be related because all the subtasks depend on the common hidden layers.

Thus training multiple related subtasks helps to obtain a stronger inductive bias among the tasks. We go on to discuss further details about the underlying mechanisms.

Data Amplification:

MTL helps to increase the sample size for training by combining multiple task datasets. Each task has independent noise in its training data. In order for each task to perform well, they have to generalize the overall noise in the dataset together. Thus, by ignoring all the noise and learning a joint representation among the shared tasks helps the model to generalize better.

Attribute Selection:

If the training data is limited and very noisy, it could be difficult for the model to differentiate between relevant and irrelevant features for solving the problem. Thus, jointly training related subtasks helps the model determine the relevant features.

Eavesdropping:

Let's say we were to train two tasks, A and B, with the hidden layer features F helpful for both tasks. But learning those hidden features F through Task B alone can be difficult due to the complex representation of Task B dataset or maybe because of the residual error in B. So, by training two subtasks together, Task B can eavesdrop the relevant features learned through A. Similarly, Task A can make use of the important features learned through Task B. Eavesdropping among the features is called catalytic hints [1].

Representation Bias:

Since the model is initialized with random weights for each run cause, the model yields different backprop nets each time. Thus, each task may generalize over different local minima. Training multiple tasks cause the model to prefer the hidden layer representation or minima, which is preferred by all tasks, i.e., the representation common among the subtasks. Also, MTL prefers to avoid that hidden representation that the individual task itself would not prefer.

Figure 2.6 shows the multitask learning architecture for four tasks. The model takes a single vector as an input while it produces four different outputs for each task. All four tasks share some hidden layers while having a task-specific layer followed by the shared layers according to the task to solve. This architecture is called a Multi-Head Architecture (MHA). Each head in the MHA solves a different task. But while training multiple related tasks together improves the accuracy among the tasks, it increases the computational cost of learning.

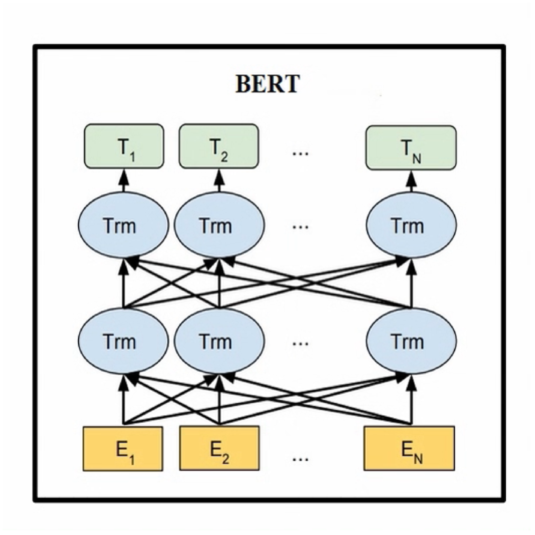

Figure 2.7: Transformers bidirectional information flow in the encoder component `https://bit.ly/37e8ZWz`

2.1.11 Encoder-only Transformer

The Transformer architecture discussed in Section 2.1.9 is an Encoder-Decoder Transformer Architecture. Transformers have also been widely used as an encoder-only [11] and decoder-only transformer [39]. In the encoder-only transformer, the output from the final attention head is passed to a linear layer instead of a decoder. Bidirectional Encoder Representations from Transformers (BERT) architecture proposed by Devlin [11] is an encoder only Transformer. BERT is trained in a multitask learning setup. The BERT model has two language heads; one for the Masked Language Model (MLM) and another for Next Sentence Prediction (NSP).

Masked Language Model

For MLM, in a given sentence, 15% of the tokens are masked with a special symbol at random, and BERT's training objective is to predict those masked tokens. To predict a masked token, the model needs to have a better understanding of the whole sentence. Unlike the decoder architecture, the encoder architecture allows the model to look into the future tokens to predict the masked word as shown in the Figure 2.7. This task is also known as the Cloze task [54].

Next Sentence Prediction

The training objective of NSP is to predict whether or not the given sentence B actually follows given sentence A 50 % of the time sentence B is the actual sentence, while 50% of the time it is a random sentence. The NSP tasks helps the model learn the correlation between the given sentences. Hence, NSP tasks help the model solve several downstream applications such as the Question Answering task, and the Entailment task.

BERT's architecture is very similar to the encoder of encoder-decoder transformer architecture. The main difference is that the output from the encoder is passed to two different fully connected layers. Thus, the model has two heads to solve the NSP and MLM tasks simultaneously. Later, to solve the different downstream tasks, only the shared transformer layers in the BERT is used with a new head, this approach is called Transfer learning. Transfer learning is a machine learning approach where the model is trained on Task A. Later, the model trained on Task A is now used to solve Task B.

2.1.12 Decoder-only Transformer

The Generative Pretrained Transformer -2 (GPT2) is a decoder only transformer proposed by Radford [39]. The GPT2 is a generative model, while the BERT we discussed in Section 2.1.11 is a discriminative model. The information flow in the encoder is bidirectional. For each token in BERT, the attention is applied to the tokens which appeared before and after it, whereas, GPT2 is autoregressive [16], as shown in the Figure 2.8 model, where each token can attend to only the previous tokens. The information flow in GPT2 is from right to left.

Unlike the decoder discussed in the Section 2.1.9, the decoder-only transformer does not have multi-head attention for the input from the encoder. The GPT2 is trained as an unsupervised language model manner on a large corpus

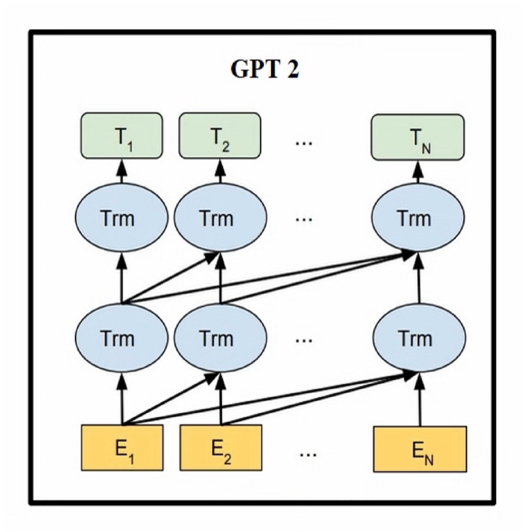

Figure 2.8: Transformers left-to-right information flow in the decoder component `https://bit.ly/37e8ZWz`

of Wikipedia and Reddit data. When given a sequence of tokens, the model predicts the next token in the sequence. Similar to BERT, the pre-trained language model GPT2 is used to solve several downstream tasks. The final layer of the GPT2 is replaced with the fully connected layer with the task-specific layers, and further fine-tuned on the downstream tasks. The advantage of using a pre-trained model reduces the data samples needed for the model to learn.

2.1.13 Generative Adversarial Network

Ian Goodfellow proposed the Generative Adversarial Network (GAN) in [15]. It is a generative model. GAN architecture has two main components: a generator model to generate new samples and a discriminator model to differentiate whether the generated sample by the generator is real or fake.

The generator G takes input noise $p_z(z)$ defined over a fixed distribution p_g. Here the generator is a multilayer perceptron. The input is sample over the distribution each time as a seed for the generator to generate output.

The discriminator D is another multilayer perceptron. The input to the

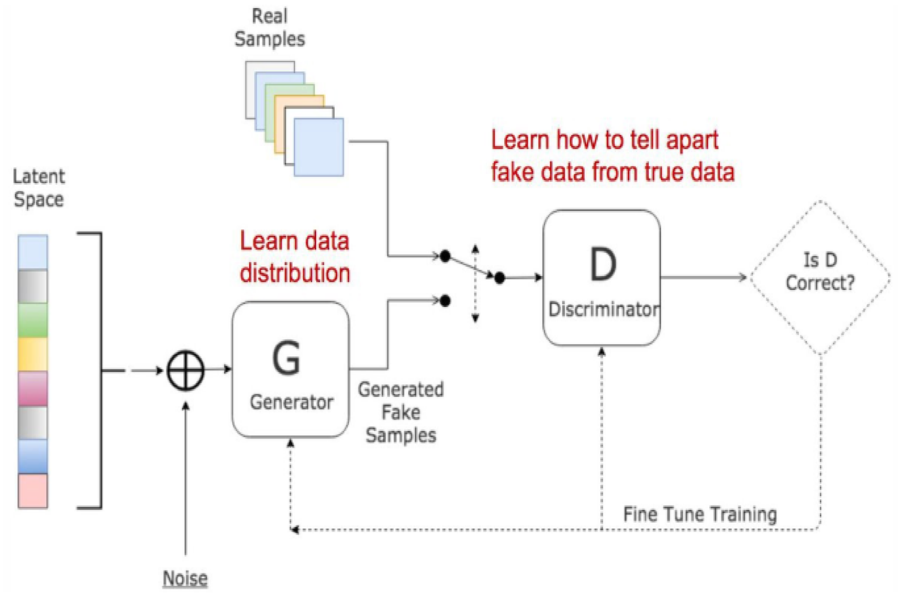

Figure 2.9: Generative Adversarial Network Architecture `https://bit.ly/3b7mNT3`

discriminator is the output from the generator, i.e., $D(G(z, \theta_g))$. The discriminator is a binary classification model, which indicates whether the generated output from the generator came from data rather than the p_g. The Figure 2.9 exhibits the architecture of GAN.

Initially, the discriminator is trained with a labeled dataset to discriminate the input from real and fake. When a generator generates a sample the discriminator assigns the correct labels. So the objective of D is to maximize the probability of assigning the correct label to the input while G is trained to minimize $log(1 - D(G))$.

$$\min G \max DV(D, G) =_{x \sim p_{data}(x)} [logD(x)] +_{z \sim p_z(z)} [log(1 - D(G(z))] \quad (2.22)$$

In the first step, the early samples generated by G are poor, and D rejects the example with high confidence as a fake sample. Over time, training G converges to generate samples to deceive D into accepting the sample as a real sample. In the second step, we stop training G, and start fine-tuning D with the samples generated G as a fake example. The GAN architecture is a minimax game between D and G, as shown in Equation 2.22; in each iteration, G tries to cheat D while D gets better in differentiating the real and fake examples.

2.2 Related Work

2.2.1 Representation learning

For any AI system to generate common-sense reasoning, it needs to understand and build a representation of the given situation [34].

Given, *You will never find a dog that likes to eat meat.*

Humans intuitively build a knowledge graph that dogs are carnivores and carnivores is a meat-eaters, so the given sentence is a false statement. Similarly, for a machine to answer questions about general situations or facts, a knowledge graph or formal logic system could be beneficial.

2.2.2 Common-sense reasoning using Formal Logic

John McCarthy proposed a system that uses formal logic [29] for common-sense reasoning. For example , $WakeUp(p)$ indicates that person p wakes up, and the fluent $Awake(p)$ indicates that person p is awake at that moment where fluent (f) means the time-varying property of the world at a timepoint; it is a boolean variable. It indicates whether the particular event (e) can occur at a given time (t) or not. So, $Awake(p)$ takes two possible values. True, if the person is awake, and false if the person is a sleep.

Based on the given axioms, we can infer that if a person wakes up, then the person will no longer be a sleep:

$$Initiates(e, f, t) \iff \exists p(e = WakeUp(p) \land f = Awake(p)) \quad (2.23)$$

If the person falls asleep, then the person will no longer be awake.

$$Terminates(e, f, t) \iff \exists p(e = FallAsleep(p)Af = Awake(p)) \quad (2.24)$$

Initiates indicates that the fluent will be true if event e occurs at time t and *Terminates* represents that the fluent will be false if the event e occurs at time t. Based on the above set of axioms, a system can make an inference that a person can sleep, or a person can be awake but not both together. So, the system can induce common-sense reasoning given all the possible axioms of that domain or about the world. However, it is not feasible to generate all the reasonable axioms about the world, but this gave rise to several other logic-based approaches for common-sense reasoning and numerous works to create logic-based ontologies, e.g., situation calculus [14], YAGO [49], DBpedia [2], and Event2mind [42].

2.2.3 Taxonomic reasoning

A taxonomy or ontology is a collection of entities, that shows the relationship between those entities and their properties. ATOMIC [45] is an ontology of the everyday activity about mental states as shown in the example 2.10.

The following statement, *X repels Y's attack* contains inferential knowledge, represented in the graph through if-then relations to distinguish cause vs. effect, and actions vs. mental states. The ATOMIC knowledge graph can be used to answer common-sense reasoning on mental states; for example, if "X repels Y's attack", it causes X to save himself from Y. The [45] used an RNN based encoder-decoder framework to train the ATOMIC dataset, where the input is an event, an if-then relation, and the target is to generate a response if the cause or effect is based on the given relationship.

ConceptNet [47] is another example of a semantic network for common-sense knowledge about practical events.

The authors of [7] took advantage of ConceptNet and the ATOMIC knowledge graph to train a decoder-only transformer model (COMET). Given two entities, COMET predicts their relationship. They represented both the graph in the triplet format, such as $entity_1, relationship, entity_2$. Given a $entity_1$ and $entity_2$, COMET generates their *relationship* given subject and object. It helps AI systems with continuous learning (like humans), as the system predicts the new relationships between the nodes in the knowledge graph based on prior knowledge. In order to predict the relationship between the entities

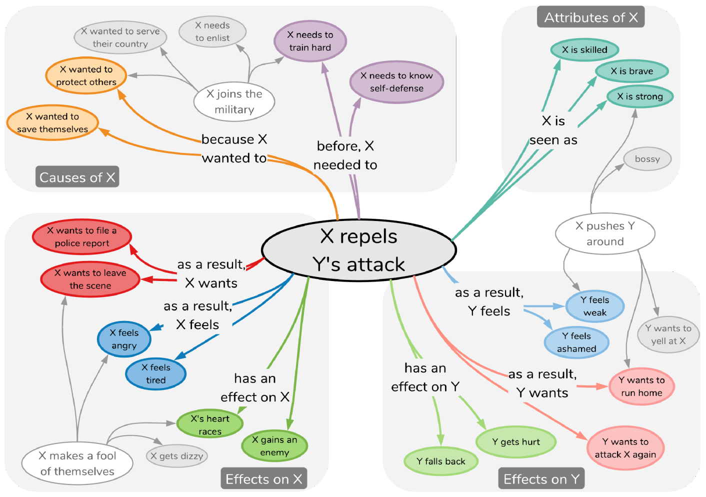

Figure 2.10: A subset of ATOMIC data set [45]

in the semantic network, the authors in COMET [7] used te ConceptNet and ATOMIC dataset. ConceptNet has also been a base for the development of many common-sense related questions answering dataset such as COMMONSENSEQA [52] and CoS-E [41]. We will discuss those datasets in the next section.

2.2.4 Inference Reasoning

Given the statement, *"The trophy would not fit in the brown suitcase because it was too big what was too big?"*, It is common sense for humans to know that it is the trophy that's too big to fit in the suitcase. Nevertheless, for a Natural Language Understanding (NLU) system, it is difficult to determine whether *it* in the statement refers to the suitcase or the trophy. The WinoGrande [44] and Winograd Schema Challenge (WSC) [25] are one of the excellent datasets for the co-reference resolution challenge. The early study by [24], [48] tried to resolve co-reference resolution using a rule-based approach. However, the recent work by [11], a model based on transformer architecture, has achieved incredible results in the co-reference resolutions task. Still, common-sense reasoning is an open and unsolved question.

The COSMOS QA [21], NarrativeQA[22], SearchQA[12] and CQA [52] are multiple-choice question answering datasets. Given text for comprehension and a question, any system would need common-sense reasoning capabilities to understand and answer the given question. Researchers have achieved competitive performance in question answering tasks requiring comprehension or a document as a source of information [41] [11] [23] [53]. In [41], the researchers use a pre-trained language model GPT2 for the generation of explanation for each multi-choice question. Later, the generated explanation is concatenated with the question and passed to the BERT. The generated explanation acts as a hint for the BERT to pick the correct choice among the plausible choices. Whereas, the authors in [11] train a transformer-based language model with the multitask objective function as CLOZE [54] task and predict whether the given two sentences are the next sentences or not.

In [56], the authors trained a language model on a vast volume of common sense dataset. Later, the model was fine-tuned for several downstream tasks. All these models help stimulate common-sense reasoning in the language model. However, they still rely on a passage as a source of information, which makes those models limited to specific domains.

Bhagavatula in [6], has proposed a task for investigating common sense reasoning through abductive reasoning. In an abductive common sense reasoning task, given two observations, the task is to generate a valid hypothesis in which both the observations are correct. The authors have achieved significant results by finetuning the pretrained decoder only GPT2 transformer model. The response generated by the language model explains when the given observation is likely to happen.

In order for an AI system to process common sense like humans, it should not depend on a source of information for each query; instead, it should learn and generate a response from previously learned information. Thus, our proposed model does not rely on any other source of information during inference other than its own prior knowledge. And unlike, [6], our task is to generate a response as to why a given statement is not possible. We train our proposed model on a wide range of common-sense datasets which helps our model reason on a wide range of topics and domains such as general science facts [31], day-to-day topics [41] [58], and a large common-sense knowledge fact-base [60].

Chapter 3

Proposed systems

The language generation task models the conditional probability of $P(Y|X)$, where X and Y are both sequences of words. In this work, we use the 36-layer decoder-only transformers based on the architecture of the Generative Pre-trained Transformer (GPT2) [39] as the backbone of the language generation task for all the presented models. The decoder only transformer architecture is similar to the original decoder transformer architecture proposed by Vaswani et al. [57]. The main difference in the case of the decoder-only transformer is that the decoder component does not have a multi-head attention for the encoder input. All our models are initialized with large weights of the pre-trained language model GPT2. In this section, we review the baseline models and the architecture of our proposed UNION model.

3.1 Language generation baseline

The main objective of our task, given a false statement as input, is to obtain an explanation as to why the given statement false. The dataset we use to train our baseline model is ComVE [58] subtaskTask C data. The subtask C dataset is formatted as of $\{X^i, \mathbf{Y}^i\}$, where each statement X^i is paired with three possible explanations $\mathbf{Y}^i = (Y_1^i, Y_2^i, Y_3^i)$. Thus, we formulate our task as sequence-to-sequence generation. We train a conditional language; it generates an explanation based on the input, i.e., the false statement. A false statement X^i is considered as the source to the language model, while one of the explanations from $\mathbf{Y}^i$ becomes the target. Therefore, as a first step, we reformat our dataset. It is no longer represented as $(X^i, Y_1^i, Y_2^i, Y_3^i)$, but rather of (X^i, Y_j^i), where $Y_j^i \in \mathbf{Y}^i$ for $j = 1, 2, 3$. Next, we train our baseline language model GPT2 with the new dataset format. The final goal is to estimate the $P(Y_j^i|X^i)$.We fine-tune a pre-trained language model as a baseline model to generate an explanation for the false statements.

3.2 Multitask Learning Baseline

We train a language model in the Multitask learning (MTL) setting and we call this model BMTL. The model has two heads, one for the language model task and another for the classification task. Raffel et al. [40], Liu et al. [27], and Devlin et al. [11] have shown that training language models using an MTL technique helps the language model to better learn and generalize.

We chose to train an MTL model using ComVE's Task B and Task C datasets because of their similarity. Task B of ComVE is a multi-choice classification problem. For each false statement X^i, there exist three plausible explanations from $\hat{\mathbf{Y}}^{\mathbf{i}} = (\hat{Y}_1^i, \hat{Y}_2^i, \hat{Y}_3^i)$. Although; only one of these is correct, each have the same syntactic structure and similar wording - only differing by a few words. Moreover, all of the correct explanations present in Task B are a subset of explanations $\mathbf{Y}$ provided in Task C (Generation). We, we hypothesized that adding this task in an MTL setting would help the model better discriminate between a valid and an invalid explanation, and consequently learn to recognize the subtle differences, thus allowing it to generate better valid reasons. Moreover, training on more similar examples to those in Task C would make the generated text more aligned with the syntactic structure and keywords of the task's data. We convert the dataset format from a multi-label to a binary classification format $(X^i, \hat{Y}_k^i, b)$, where b is a binary label (correct or not), and $k = 1, 2, 3$. The Tasks B and C have the same set of false statements X as input. Thus, we pair the dataset according to the false statement. The new data format is the following: $\{(X^i, Y^i_j), (X^i, \hat{Y}^i_k, b)\}$. We train the language modeling and the classification heads in parallel. The loss is computed by taking the summation of both heads' losses.

In multitask learning, the model takes a single input to the model while it has several heads to solve different subtasks simultaneously. Therefore, another major issue in training the multitask learning with language generation and classification task simultaneously is the input format to the model. The language model takes only X, whereas the classification head takes $X + \hat{Y}$. So, we combine both the tensors the $[[c + X][c + X + \hat{Y}]]$ generate response as well as a classification label for both tensors. But during loss computation, we ignore the language head loss for the classification tensor and the classification loss for the language tensor. It helps us train the classification and language generation simultaneously.

3.3 RAML

We proposed the Baseline + REINFORCE model for common-sense response generation, we call this model RAML. Our work was inspired by GAN architecture [15] and REINFORCE algorithm [13] [51].

$$L(\theta) = -\sum_{(j=1,|V|)} y^t_j log(\hat{y}^t_j) + (-(blog(\hat{b}) + (1-b)log(1-\hat{b}))) \tag{3.1}$$

$$L_{RAML}(\theta) = L(\theta) * P_{EA}(X, \hat{y}) \tag{3.2}$$

The architecture of the RAML model is closely related to the architecture of the BMTL model. But the loss function for the LM head of the RAML model is different from that of the BMTL. In Equation 3.1, the first part of the equation is the language head loss; the second part of the equation 3.1 is the classification loss. Equation 3.2 is the RAML model loss, where the multitask learning equation is multiplied with the reward generated by the Estimated Approval.

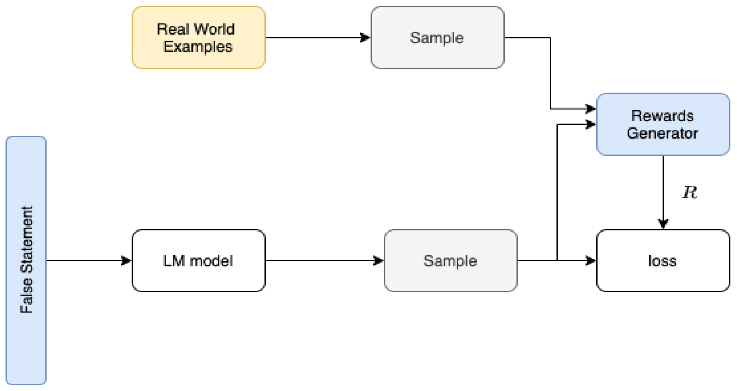

Figure 3.1: RAML Architecture

Estimated Approval (EA) is a discriminative model. It is a binary classifier. Given two statements, A and B, the EA model generates a soft score, which indicates whether B is an explanation of the given false statement A. The EA is trained on the ComVE Task B and C dataset. As explained in Section 3.2, Task B is a multi-choice classification dataset, and Task C is the multiple explanations for the false statement. Since all the explanations in Task C are valid, we label all the explanations as valid, i.e., (X^i, Y^i_j, b) where b is 1. But for Task B $(X^i, \hat{Y}^i_k, b)\}$, the b can be 1 or 0, where $\mathbf{\hat{Y}^i} = (\hat{Y}^i_1, \hat{Y}^i_2, \hat{Y}^i_3)$.

Unlike GAN architecture, the discriminator is not trained along with the generator iteratively. Instead, it is fine-tuned once before training the generator. The generator is the language model. During training, given a source statement X^i and the generated target statement, $\hat{Y}$ is concatenated and given as an input to the EA. The soft-score of the EA is that the probability of B is not a valid explanation to A. In the initial training, the soft-score is high as the explanation generated by the language model becomes more accurate as the soft-score becomes zero, and the training is stopped.

During inference, we ignore the classification head and the discriminator, and we generate an explanation from the language head.

3.4 Why UNION?

A language model is a probability distribution learned over the sequence of tokens in the training corpus. Thus, the performance of the model strongly depends on the quality of the data used for its training. When we investigated the explanations generated by the baseline and MTL models, it seemed that these tended to often negate the given input statement, for example refer to the 4.11. Although this may still make them valid explanations, it did not necessarily make them express reasoned explanations about why the input is a false statement. Part A of Table 4.11 provides examples of two valid explanations where the first is just a negation of the input whereas the second is a better systematic explanation of why the given statement is false. The root cause for generating a simple negation of a statement over a reasoned explanation is the ComVE dataset itself. More than one-third of it contains negating statements, which most probably signals to the model to learn to negate any given input. To deal with this issue, we may either remove the explanations that negate to the false statement or increase the dataset by adding better explanations. Since the dataset for explanation generation is limited, deleting any explanations may have a negative impact, while creating a new dataset is a tedious task in itself. Therefore, we resorted to using other common-sense related datasets, CoS-E, OpenBook, and OMCS , along with the ComVE dataset, to train the language model to generate better responses. These additional datasets treat different issues. OpenBook is a question answering dataset related to science facts. CoS-E is for general common-sense question answering. OMCS contains common knowledge facts. Training all four together leads to two main difficulties or questions to answer:

1. Each dataset is related to a different issue while our main task is to generate an explanation for a false statement. How can we force the model to generate an explanation response that is specific to the false statement and not just to any random generic statement related to it?
2. Each dataset has a different number of classification choices. How do we train all of them together in an MTL setting?

Our proposed architecture, described in Section 3.5, solves the first problem with the help of a contextual keyword and the second by merging all the classes.

False Statement	Explanation
A) The chocolate cried.	1) Chocolate doesn't cry. (Baseline)
	2) Chocolate is an inanimate, non human thing and cannot cry. (UNION)
B) Sugar is used to make coffee sour.	1) Sugar is used to make coffee sweet. (UNION)
	2) Sugar is not used to make coffee sour. (Baseline)

Table 3.1: Examples from ComVE explanation (Generation), Example A is from the baseline model and B is from the UNION model

3.5 UNION

The backbone of the mUltitask learNIng for cOmmonsense reasoNing (UNION) architecture in Figure 3.2 is a decoder-only transformer.

The UNION model has four major layers: shared, pre-trained, semi-shared, and task-specific.

The first layer is a shared layer. It is the base layer and shared layer for the remaining layers.

The UNION model is trained in a multitask learning structure. The multitask learning defined in the previous section 2.1.10 uses a different head to solve each task in parallel, whereas in the UNION architecture, we use a single language model head to train all three response generation tasks, i.e., ComVE [58], CoS-E [41], and Openbook [31].

The model generates an explanation $\hat{Y}$ for ComVE, CoS-E, and Openbook dataset given a conditional context as input X. In Equation 3.3, X is the input to the model, and y is the response.

$$P(Y|X) = \prod_{t=1}^{n} P(y_t|X, y_1, y_2, ..., y_{t-1}) \tag{3.3}$$

On the other hand, the OMCS [60] dataset has over a million statements as facts and common knowledge, e.g., "You are likely to find a shelf in a cupboard". The source and target for the OMCS dataset are the X. The

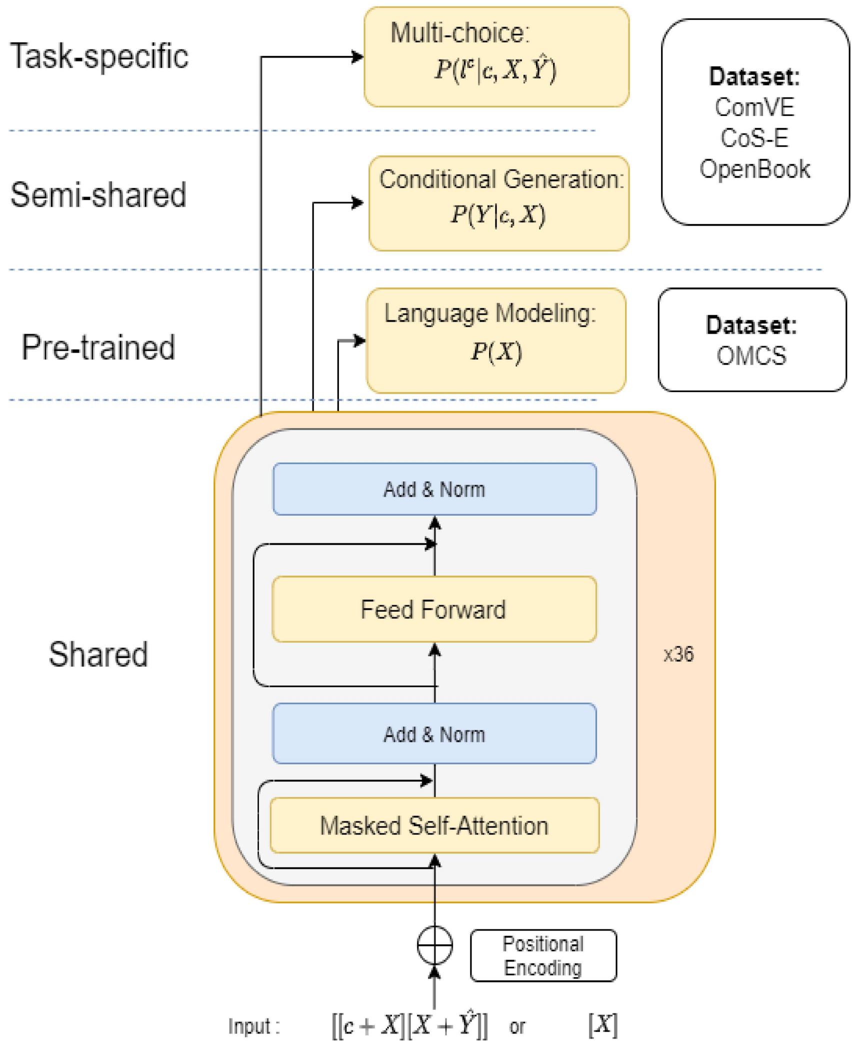
Task-specific
Multi-choice: $P(l^c|c, X, \hat{Y})$
Dataset: ComVE CoS-E OpenBook
Semi-shared
Conditional Generation: $P(Y|c, X)$
Pre-trained
Language Modeling: $P(X)$
Dataset: OMCS
Shared
Add & Norm
Feed Forward
Add & Norm
Masked Self-Attention
x36
Positional Encoding
Input : $[[c + X][X + \hat{Y}]]$ or $[X]$

Figure 3.2: UNION Architecture

as the OMCS head predicts the next token in the sequence given a token, as shown in 3.4.

$$P(X) = \prod_{t=1}^{n} P(x_t|x_1, x_2, ..., x_{t-1}) \tag{3.4}$$

Thus, it makes it an obvious choice for training the OMCS as a separate head rather than training it along with other explanations. The OMCS dataset is general common knowledge about science and psychology. Thus, we pre-train our language model head with the OMCS dataset. During pre-training, we set the loss of other heads to zero. The pre-training helps the model learn general facts and knowledge about the world. As a reason, we define a separate layer called a pre-trained layer for the OMCS dataset. While the ComVE, OpenBook, and CoS-E share the same layer called a semi-shared layer.

Even though the semi-shared layer is a conditional language head, the explanation generated is not coherent to the given task. For example, given statements such as, "*He ate a tomato from the dirt*", from the ComVE dataset it is difficult even for humans to classify it as a false statement and explain why it is a false statement. Thus, the response generated by the model during inference is an explanation that is not related to the ComVE task but more of a general explanation because of the OpenBook and CoS-E dataset influence. But the main hypothesis of our research regards the non-sensical statement and generation of a response that explains why that given statement is nonsensical. As per our hypothesis, the desired explanation is, "*nobody would eat a tomato from the dirt.*" In order to solve this problem, we use a contextual keyword.

$$P(Y|c, X) = \prod_{t=1}^{n} P(y_t|c, X, y_1, y_2, ..., y_{t-1}) \tag{3.5}$$

The contextual keyword c is a special token that signals the model indicating the task to generate an explanation that is coherent to the given task. We use a different contextual keyword for each dataset in the semi-shared multi-head, i.e., "cose" for CoS-E, "comve" for ComVE, and "openbook" for OpenBook. The contextual keywords are case sensitive. Each keyword is not part of our training vocabulary. Thus it gives a unique signal to the model indicating what type of explanation is expected as an output. During inference, we only use the contextual keyword "comve" since the given input to the model is always a nonsensical statement. Though the layer is complete it is shared among different tasks, and the explanation generation also depends on the contextual keyword; hence we call it a semi-shared layer.

The final layer in the UNION model is a multi-class classification layer. Unlike the language model head, it is not possible to use a single classification layer for all the tasks as the number of labels is different. The ComVE tasks have three plausible explanations for each, whereas OpenBook has four, and CoS-E has five . So, we have three heads for the multi-choice, where each head solves the independent multi-choice classification. During training based on the contextual keyword, the UNION model triggers the respective classification head.

The limitation of the UNION model it assumes as the given input is always a nonsensical statement. If a valid statement is given as an input, then the UNION model generates a negation of the valid statement. Even a false statement can be true given valid context; for example, "He was starving to death. Hence he ate a tomato from the dirt." The UNION model takes a single statement as input, ignoring the context before and after the given statement.

3.6 Common-sense System

The primary assumption of the UNION model is that the given input is a false statement. In order to incorporate our common system into a dialogue system, it is essential to classify whether or not the given statement is sensical. We train a Statement Discriminator (SD), as a discriminative model. It classifies whether the given statement is sensical or nonsensical. If the given statement is sensical, then the follow of information in the dialogue system is unchanged,whereas if the SD classifies the statement as nonsensical, then the input is passed to the UNION model. The UNION generates a response that explains why the given the input statement is nonsensical. The SD is an

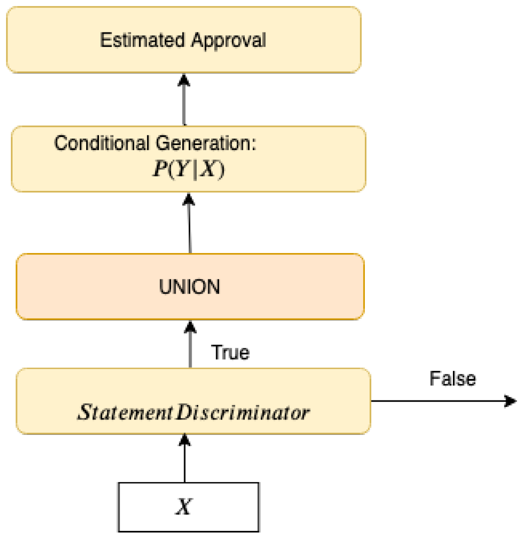

Figure 3.3: Dialogue flow for a common-sense system. During inference the contextual keyword is fixed as "comve" hence we haven't represented c in the figure

encoder-only transformer BERT, as discussed in Section 2.1.11.

We use Task A of the ComVE dataset to train the SD model. The Tasks A of ComVE data is a binary classification dataset. Each data sample has two statements; statements A and B, and a binary label to indicate which statement is a valid explanation (A^i, B^i, b^i). If b is zero, then statement A is a valid statement otherwise statement B is valid. To classify a statement as sensical or nonsensical in a dialogue system, the classifier needs to predict using a given single statement. So, we translate the dataset to $\{ (A^i, b^i), (B^i, 1 - b^i) \}$, i.e. $X \in \cup(A, B)$.

In the new data format, b indicates whether the given statement is valid or not. Now, the SD estimates the $P(b|X)$ and not the $P(b|A, B)$ as in the original data format. There is an infinite number of false statements possible using a different combination of tokens in a vocabulary. Thus, we use the ED from Section 3.3 to classify whether the generated response makes sense or not. If the generated response does not make sense, then the control is given to the dialogue to handle the respective dialogue flow. The entire flow of the pipeline system is shown in the Figure 3.3

Chapter 4

Experiments and Results

In this chapter, we discuss in detail the various datasets we used in this study, and we evaluate our UNION model with the different evaluation metrics we proposed. We compare the UNION model against two baseline models Baseline and Baseline+MTL. We evaluate each model for different measures such as BLEU, perplexity for fluency of the generated response, Estimated Approval, and Uniqueness for the acceptance of diverse responses. Finally, we investigate the significance of each dataset for commonsense response generation by ignoring the particular data set during training.

4.1 Datasets

4.1.1 Common Sense Validation and Explanation

The primary dataset we use for common sense explanation generation is Common sense Validation and Explanation (ComVE)[58]. The ComVE dataset has three tasks in it. The first task is binary classification; given two statements with similar wordings, it must choose which makes sense and which does not.(For examples, refer to Table 4.1.) The second task is the multi-choice classification, as shown in Table 4.2. This finds the best possible explanation from the three options that explains why the given statement does not make sense. The third task is a response generation task shown in Table 4.3; given a false statement, the model needs to generate an explanation as to why the given statement does not make sense.

Input	Label
A) He wrote an exam in pen B) He wrote an exam in knife	A

Table 4.1: Binary Classification

Question	Choices	Label
He wrote an exam in knife	A) Knife is very sharp to kill the exam B) Knife is stainless but the exam is usually white C) Knife is not supported to write in paper	C

Table 4.2: Multi-Choice Classification

False Statement	Possible Explanation
He wrote an exam in knife	A) Knife is not supported to write in paper. B) No one can write with a knife. C) Knives are not writing utensils.

Table 4.3: Explanation Generation

4.1.2 Open-book Question Answering

The open-book question [31] answering is an elementary level science fact question answering dataset that probes common knowledge understanding of the system. It is a multi-choice classification dataset. Each question has four possible explanations and a science fact associated with the question that indirectly helps the system pick the best viable option from the question. Refer to Table 4.4 for examples.

Question	Choices	Label
Which of these items would let the most heat travel through?	A) a new pair of jeans. B) a steel spoon in a cafeteria. C) cotton candy at a store. D) a Calvin Klein cotton hat.	B
Science Fact	Metal is a thermal conductor	

Table 4.4: Open book Question Answering

4.1.3 Common-Sense Explanation

The Common sense Explanation (CoS-E) [41] dataset is a multi-choice question answering dataset with a human explanation for each question. The CoS-E dataset is an extension of the Common sense QA [52] dataset, where the initial dataset did not have the human explanation for each question. Table 4.5 is an example of the CoS-E dataset.

Question	Choices	Label
If a person doesn't have pants that fit, what should he do?	A) buy a monkey B) let himself go C) buy clothes D) bank money E) catch a cold	C
Explanation	new pants must be bought	

Table 4.5: CoS-E Example

4.1.4 Open Mind Common Sense

The Open Mind Common Sense (OMCS) [60] dataset is a large collection of common sense knowledge. It has more than a million facts in English. The OMCS dataset has different types of common-sense-based statements. Some facts convey the relationships between objects and events, while some statements contain information about the emotional context of situations. For example, "A coat is used for keeping warm ", "The last thing you do when you cook dinner is wash your dishes", "Spending time with friends causes happiness", etc.

		Training Data size	Dev size	Test size
ComVE	LM	30,000	997	1,000
ComVE	MC	10,000	997	1,000
CoS-E	LM	6819	1948	974
CoS-E	MC	6819	1948	974
Open book	LM	4957	500	500
Open book	MC	4957	500	500

Table 4.6: Dataset Statistics

The Table 4.6 summarizes the statistics of the dataset size used for training. The ComVE, CoS-E and Open-book was divided into train, dev and test split. While the OMCS dataset was used only to pretrain the model while no dev or test split was carried for that.

4.2 Evaluation

4.2.1 BLEU and Human Evaluations

The BLEU score [35] is widely used in tasks such as machine translation [50] [9], which calculates the overlap between the candidates and the reference text. However, similar to dialogue generation, we found it is not an ideal measurement because it does not tolerate diversity. For example, as shown in Table 4.7, "Book is not a time keeping device." has little overlap with the reference but it should still be considered a good generation. To remedy this, the submitted systems are evaluated by human evaluations.

The human evaluation score is based on the agreement between three reviewers. The evaluation is carried out on a subset of 100 random samples of the test set. The rubrics are as shown below:

0 - If the generated reason is grammatically incorrect or the reason itself does not make sense.

1 - If the generated explanation is just a negation of the given statement.

2 - If the generated reason is relevant to the given false statement, but there are grammatical errors or irrelevant explanations in it, too.

3 - If the generated reason is a valid explanation with no grammatical errors.

The BLEU score and human evaluation are carried out by the Sem-Eval task organizers. Our proposed model achieves a human evaluation score of 2.10, highest human evaluation score obtained in the ComVE dataset in the Sem-Eval task. However, human evaluations are very expensive and risk being subjective.

False Statement	We use book to know the time
Referential Reasons	a) A book is used to study b) A book does not have the ability to show what time it is. c) A books doesn't tell the time
Generated Explanation	Book is not a timekeeping device.

Table 4.7: UNION Model Generated Explanation

4.2.2 Additional automatic metrics

In order to better evaluate the proposed systems and conduct ablation studies, we additionally measure the quality of the explanation generated by our models using several auxiliary automatic metrics.

Models	BLEU	PPL - Gen.	PPL - Trg.	EA	UNI	Length
Baseline	10.36	970.05	495.35	0.86	3.55 ± 1.77	5.5 ± 1.97
Baseline + MTL	12.4	357.59	331.22	0.93	3.31 ± 1.68	5.59 ± 1.89
Baseline + REINFORCE	10.70	229.99	332.83	0.90	6.81 ± 2.38	4.87 ± 2.29
UNION + No CoSE	13.28	62.89	238.64	0.96	5.82 ± 2.10	8.51 ± 2.08
UNION + No Open book	13.75	142.19	260.38	0.95	4.29 ± 1.87	6.46 ± 2.19
UNION + No OMCS	15.7	194.66	243.83	0.94	4.29 ± 1.79	6.41 ± 2.06
UNION	16.36	135.1	212.1	0.97	4.53 ± 2.05	6.59 ± 2.3

Table 4.8: Results on an average of 1000 test samples from the ComVE task

Model	Precision	Recall	F1-Score
SD - Base	0.8679	0.8583	0.8631
SD - Large	0.8752	0.8604	0.8678
SD - Random	0.359	0.4927	0.4258

Table 4.9: Estimated Approval Classifier Results

Model	Precision	Recall	F1-Score
EA - Base	0.8258	0.7477	0.7867
EA - Large	0.8199	0.7715	0.7957
EA - Random	0.4284	0.4994	0.4639

Table 4.10: Statement Classifier Results

Perplexity:

We first use perplexity to measure the grammatical correctness of the generated text. Particularly, we use two variants of perplexity: general perplexity (*ppl-gen*) and target corpus perplexity (*ppl-trg*). We measure *ppl-gen* by using GPT-2 LM head directly, as it is pre-trained on large scale corpus (reddit, wiki). It gives an indicator of how fluent the generated content is in general. In the meantime, we also want to measure how well the generations can in terms of fitting into the dialect of the target corpus. We achieve this by training an n-gram language model with Kneser-Ney smoothing[1].

Informative:

We observe that the generations by the baseline models are often are derived from X by changing very few words, as shown in Table 4.11. Therefore, we propose two auxiliary metrics: *Estimated Approval (EA)* and *Uniqueness (UNI)*.

UNI metrics measure how much more information is given by the answer; it calculates the number of tokens that are not present in the given input. A higher UNI value suggests more diverse and potentially more informative

[1]https://github.com/kpu/kenlm

False Statement	Possible Explanation
The room was dark, so I turned on the stereo.	A) A stereo does not provide illumination. B) A stereo does not provide illumination. C) A stereo does not make light.

Table 4.11: Issues with ComVE Generation Task dataset

keywords are used. On the other hand, an entirely irrelevant text may also achieve a high UNI score. Therefore, in addition to UNI, we use EA described in Section 3.3.

Moreover, we evaluate the generated explanation by the length of the responses. The length metric helps us to understand the average length of the generated response. Similar to UNI, a random set of tokens can help to get a better length score. Thus, it is crucial to have the right balance between the UNI and the length score.

4.3 CAGE Framework:

The CAGE framework proposed by **cose**, for multi-choice classification, has achieved state-of-the-art results in the CoS-E dataset. The CAGE framework consists of a language model (GPT2) and a discriminator (BERT) for classification. The language model is trained to generate an explanation given a question and multiple-choice answers. Later, the explanation generated by the LM is appended to the input. The new data is in the format of a question, explanation, and multiple-choice answer. The new format is passed as an input to the discriminator for classification.

By replacing the LM model with the UNION, the model shows incredible results in the classification task. Table 4.12 summarizes the results of the UNION model and other related models. Unlike the CAGE language model, the UNION model only takes the question as an input and not the multiple-choice answer. Only, the language model is replaced with the UNION model, while the remaining of the framework remainder the same as CAGE. The generated response is appended to the question along with the multiple-choices

Method	Accuracy (%)
RC [52]	47.7
GPT2 [52]	54.8
CoS-E open-ended [41]	60.2
CAGE reasoning [41]	64.7
CAGE + UNION	74.12
Humans [52]	95.3

Table 4.12: CAGE + UNION results

answers.

4.3.1 F1 Score:

The dataset used to train the SD and EA is class imbalanced, so the cost of each label is different. Hence we report F1-score, Precision, and Recall for each model over accuracy.

$$P = \frac{TruePositive}{TruePositve + FalsePositive} \tag{4.1}$$

$$R = \frac{TruePositive}{TruePositve + FalseNegative} \tag{4.2}$$

$$F1 = 2 * \frac{Recall * Precision}{Recall + Precision} \tag{4.3}$$

P is the precision, R is the recall, and $F1$ indicates the F1-score, which is the weighted average of Precision and Recall.

4.4 Analysis

Table 4.8 summarizes the results obtained by the various models that we have tried for this task. The difference between *UNION - No CoS-E*, *UNION - No OpenBook*, *UNION - No OCMS*, and *UNION* is only the training datasets,

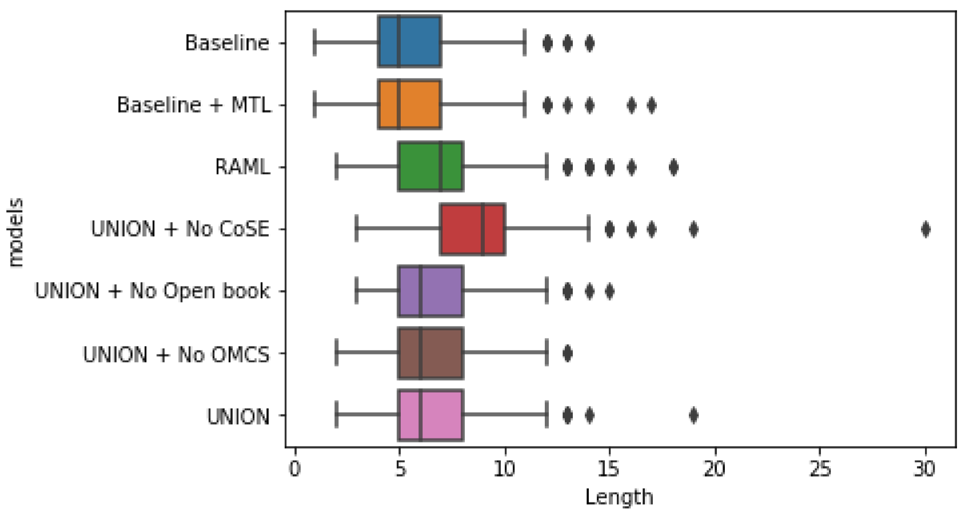

Figure 4.1: The above figure shows the varying length of explanation generated by each model.

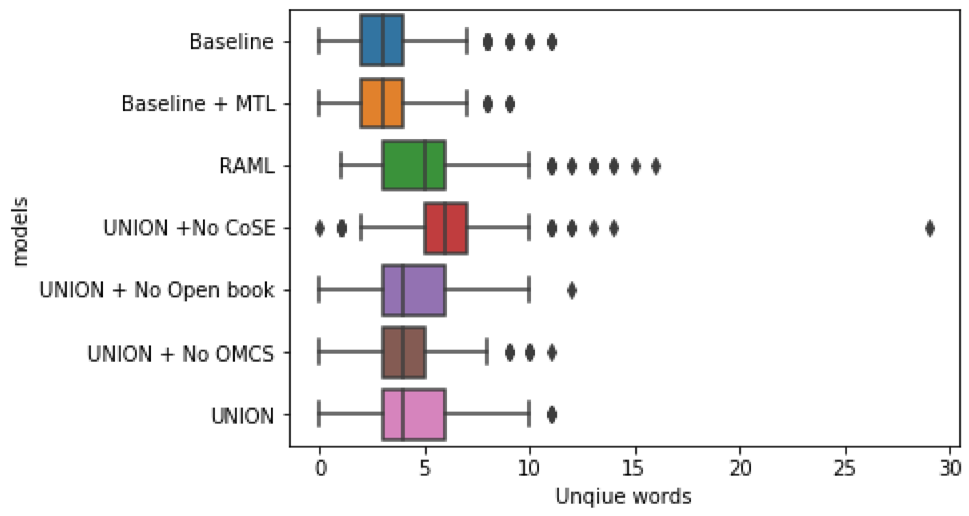

Figure 4.2: The above figure shows the unique words in the generated explanation, which are not part of the given statement.

while the architecture for all the models remains the same. The - *No* stands for without. It indicates that a particular dataset was ignored while training the UNION model. The perplexity, EA score, and the average length of the generations by *UNION - No CoS-E* are better than *UNION - No OpenBook*.

It is important to note that the CoS-E dataset is open-ended, common sense question answering. The ComVE dataset, however, has been constructed by annotators who were influenced by ConceptNet to generate false statements. ConceptNet is a semantic graph of commonsense knowledge developed from the OMCS dataset. Thus, training the UNION model with the OMCS dataset leads to better results than the *UNION - No OCMS model*. The OpenBook dataset is related to scientific facts, which is similar to the OMCS dataset. Thus, the *UNION - No CoS-E* performs better than *UNION - No OpenBook* while the UNION model performs better than all the other models. The initial model we submitted to the SemEval2020 Task 4 is *UNION - No OCMS*. It achieved a BLEU score of 15.7 and a human evaluation score of 2.10 (Ranked 1st). Later, when training the model with the OMCS dataset, the BLEU score of the model increased by 0.7. Examples of generated explanations from all models are provided in Appendix A.1.

The boxplot in Figures 4.1 and 4.2 represents the average size of the explanation and the average number of unique words in the explanation over 1000 sentences. The higher the length of the explanation, the higher the probability of having unique words in the explanation. In the average size boxplot and unique words boxplot the explanation generated by the UNION + No CoSE has higher value compared to other models. It is because the average length of the explanations and the multi-choices in the CoS-E dataset is short, and consequently it hurts the explanation length for UNION. Yet, we include the COSE dataset while training the UNION model because it improves the quality of the explanation that is evaluated by the EA and even the model receives a better BLEU score. The dots in both the boxplot represent the outliers. Whereas, the center line inside the box represents the median value of the average length and average number of unique words in the explanation.

Chapter 5

Conclusion

In this thesis, we have addressed the problem of common sense for the conversational agent. The conversational agent recently has been very successful. However, the most open domain model lacks in common sense knowledge. For a chatbot to have a human-like conversation, chatbots need to have common sense. We have introduced a common-sense system devoted to common sense reasoning. For a chatbot to stimulate and have a conversation like humans, it needs to detect and generate reasoning only for those inputs that do not make sense or the question it has been asked.

In this research, we proposed an end-to-end common sense reasoning system. The statement discriminator model helps to detect whether the given statement makes sense or not. The UNION model is used to generate an explanation for the given non-sensical statement. This architecture helps to plug our proposed system into any of the pre-defined chatbot systems with ease.

At times the generated explanation by an end-to-end model itself may not make sense or may not be coherent with the given input. To solve this problem, we use an estimated approval model that evaluates the generated explanation by the UNION model during runtime. If the generated explanation is not good enough, we can rely on the default conversational agent.

To train the UNION model, we use four different common sense related data set. We primarily use the ComVE for a common-sense explanation generation. Based on the human evaluation score, we achieve state-of-the-art results on the ComVE dataset.

As human evaluation can be expensive and BLEU score for the conversational task, especially for reasoning tasks, could be an inaccurate metric for comparison of models. Thus, we propose automatic auxiliary metrics for the comparison of our proposed models.

For future work, we would like to investigate further in the following areas:

1. The UNION model we defined, is common-sense reasoning based on the general facts about the world. It does not consider the domain knowledge for common reasoning, where domain knowledge can be user-specific details. Humans learn about other humans through conversation and generate a response based on the learned knowledge during the conversation. In the future, we would like to generate common sense based response by continuous learning through conversation.

2. We would also like to work on answering questions related to common sense rather than just generating an explanation to the false statement. The question answering could be general knowledge questions, or it could be a simple mathematical problem or generating a complicated explanation based on the quantitative number from a database.

References

[1] Y. S. Abu-Mostafa, “Learning from hints in neural networks,” *Journal of complexity*, vol. 6, no. 2, pp. 192–198, 1990.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, “Dbpedia: A nucleus for a web of open data,” in *The semantic web*, Springer, 2007, pp. 722–735.

[3] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.

[4] J. Baxter, “A bayesian/information theoretic model of learning to learn via multiple task sampling,” *Machine learning*, vol. 28, no. 1, pp. 7–39, 1997.

[5] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[6] C. Bhagavatula, R. L. Bras, C. Malaviya, K. Sakaguchi, A. Holtzman, H. Rashkin, D. Downey, S. W.-t. Yih, and Y. Choi, “Abductive commonsense reasoning,” *arXiv preprint arXiv:1908.05739*, 2019.

[7] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz, and Y. Choi, “Comet: Commonsense transformers for automatic knowledge graph construction,” *arXiv preprint arXiv:1906.05317*, 2019.

[8] R. Caruana, “Multitask learning,” *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[9] D. Chiang, “A hierarchical phrase-based model for statistical machine translation,” in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, 2005, pp. 263–270.

[10] J. L. Cummings, D. F. Benson, M. A. Hill, and S. Read, “Aphasia in dementia of the alzheimer type,” *Neurology*, vol. 35, no. 3, pp. 394–394, 1985.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[12] M. Dunn, L. Sagun, M. Higgins, V. U. Guney, V. Cirik, and K. Cho, "Searchqa: A new q&a dataset augmented with context from a search engine," *arXiv preprint arXiv:1704.05179*, 2017.

[13] S. Edunov, M. Ott, M. Auli, D. Grangier, and M. Ranzato, "Classical structured prediction losses for sequence to sequence learning," *arXiv preprint arXiv:1711.04956*, 2017.

[14] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[15] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[16] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[17] D. Gunning, "Machine common sense concept paper," *arXiv preprint arXiv:1810.07528*, 2018.

[18] M. Henderson, B. Thomson, and J. D. Williams, "The second dialog state tracking challenge," in *Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue (SIGDIAL)*, 2014, pp. 263–272.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] L. Holmstrom and P. Koistinen, "Using additive noise in back-propagation training," *IEEE transactions on neural networks*, vol. 3, no. 1, pp. 24–38, 1992.

[21] L. Huang, R. L. Bras, C. Bhagavatula, and Y. Choi, "Cosmos qa: Machine reading comprehension with contextual commonsense reasoning," *arXiv preprint arXiv:1909.00277*, 2019.

[22] T. Kočisk, J. Schwarz, P. Blunsom, C. Dyer, K. M. Hermann, G. Melis, and E. Grefenstette, "The narrativeqa reading comprehension challenge," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 317–328, 2018.

[23] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, *et al.*, "Natural questions: A benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, 2019.

[24] H. Lee, A. Chang, Y. Peirsman, N. Chambers, M. Surdeanu, and D. Jurafsky, "Deterministic coreference resolution based on entity-centric, precision-ranked rules," *Computational Linguistics*, vol. 39, no. 4, pp. 885–916, 2013.

[25] H. J. Levesque, E. Davis, and L. Morgenstern, "The winograd schema challenge," in *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012*, 2012. [Online]. Available: `http://www.aaai.org/ocs/index.php/KR/KR12/paper/view/4492`.

[26] C. S. Lewis, "Studies in words / by c. s. lewis.," *Cambridge London : Cambridge U.P*, 1967.

[27] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," *arXiv preprint arXiv:1901.11504*, 2019.

[28] X. Liu and W. B. Croft, "Statistical language modeling for information retrieval," *Annual Review of Information Science and Technology*, vol. 39, no. 1, pp. 1–31, 2005.

[29] J. McCarthy, *Programs with common sense.* RLE and MIT computation center, 1960.

[30] M.-M. Mesulam, "Slowly progressive aphasia without generalized dementia," *Annals of Neurology: Official Journal of the American Neurological Association and the Child Neurology Society*, vol. 11, no. 6, pp. 592–598, 1982.

[31] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, "Can a suit of armor conduct electricity? a new dataset for open book question answering," *arXiv preprint arXiv:1809.02789*, 2018.

[32] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[33] T. Mikolov, M. Karafiát, L. Burget, J. Černock, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[34] E. T. Mueller, *Commonsense reasoning: an event calculus based approach.* Morgan Kaufmann, 2014.

[35] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2002, pp. 311–318.

[36] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013, pp. 1310–1318.

[37] A. Perumal, C. Huang, A. Trabelsi, and O. R. Zaıane, "Ana at semeval-2020 task 4: Multi-task learning for commonsense reasoning (union)," 2020.

[38] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[39] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[40] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019.

[41] N. F. Rajani, B. McCann, C. Xiong, and R. Socher, "Explain yourself! leveraging language models for commonsense reasoning," *arXiv preprint arXiv:1906.02361*, 2019.

[42] H. Rashkin, M. Sap, E. Allaway, N. A. Smith, and Y. Choi, "Event2mind: Commonsense inference on events, intents, and reactions," *arXiv preprint arXiv:1805.06939*, 2018.

[43] A. Ratnaparkhi, "A maximum entropy model for part-of-speech tagging," in *Conference on Empirical Methods in Natural Language Processing*, 1996.

[44] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, "Winogrande: An adversarial winograd schema challenge at scale," *arXiv preprint arXiv :1907.10641*, 2019.

[45] M. Sap, R. Le Bras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N. A. Smith, and Y. Choi, "Atomic: An atlas of machine commonsense for if-then reasoning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3027–3035.

[46] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[47] R. Speer, J. Chin, and C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[48] V. Stoyanov and J. Eisner, "Easy-first coreference resolution," in *Proceedings of COLING 2012*, 2012, pp. 2519–2534.

[49] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 697–706.

[50] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[51] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[52] A. Talmor, J. Herzig, N. Lourie, and J. Berant, “Commonsenseqa: A question answering challenge targeting commonsense knowledge,” *arXiv preprint arXiv:1811.00937*, 2018.

[53] Y. Tay, L. A. Tuan, and S. C. Hui, “Multi-range reasoning for machine comprehension,” *arXiv preprint arXiv:1803.09074*, 2018.

[54] W. L. Taylor, “
cloze procedure”: A new tool for measuring readability,” *Journalism quarterly*, vol. 30, no. 4, pp. 415–433, 1953.

[55] C. Tillmann, “A unigram orientation model for statistical machine translation,” IBM THOMAS J WATSON RESEARCH CENTER YORKTOWN HEIGHTS NY, Tech. Rep., 2004.

[56] T. H. Trinh and Q. V. Le, “A simple method for commonsense reasoning,” *arXiv preprint arXiv:1806.02847*, 2018.

[57] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[58] C. Wang, S. Liang, Y. Zhang, X. Li, and T. Gao, “Does it make sense? and why? a pilot study for sense making and explanation,” *arXiv preprint arXiv:1906.00363*, 2019.

[59] S. Young, M. Gašić, B. Thomson, and J. D. Williams, “Pomdp-based statistical spoken dialog systems: A review,” *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1160–1179, 2013.

[60] C.-H. Yu and H.-H. Chen, “Commonsense knowledge mining from the web,” in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Appendix A

Apendix

A.1 Examples of generations for different models

False statement : The Pokemon caught Ash

Baseline : a pokemon is not a bird

Baseline + MTL :pokemon are not animals

UNION + No Cose : a pokeball is used to catch a pokemons

UNION + No OpenBook : pokemon are not alive

UNION + No OCMS : pokemon are not alive

UNION : Ash is a human and cannot be caught by a pokemon.

False statement : You should iron your clothes before you wash them

Baseline : ironing clothes is not the same as washing

Baseline + MTL :ironing clothes is not a good thing to do

UNION + No Cose : ironing clothes before washing them will make them dirty.

UNION + No OpenBook : ironing clothes before washing them will make them dirty.

UNION + No OCMS : ironing your clothes before washing them will make them cleaner

UNION : you should wash your clothes before you iron them

False statement : I was on date with lion yesterday.

Baseline : lion was not on date with girl

Baseline + MTL :lion is not a date.

UNION + No Cose : lion is not a human to be on date with.

UNION + No OpenBook : lion is a wild animal and cannot be on a date.

UNION + No OCMS : lion is not a human to date.

UNION : Lions are wild animals and are not meant to be on dates.

False statement : He broke his leg and went to get a burger

Baseline : burgers are not dangerous

Baseline + MTL : burgers are not broken.

UNION + No Cose : you don't get a burger from breaking your leg.

UNION + No OpenBook : a burger is not a medical treatment

UNION + No OCMS : burgers are not used to heal broken legs.

UNION : Break your leg and you will get a cast.

False statement : People use cars to fly.

Baseline : cars are not the most efficient

Baseline + MTL :cars are not used for flying

UNION + No Cose : cars don't have the ability to fly.

UNION + No OpenBook : cars don't fly.

UNION + No OCMS : cars cannot fly.

UNION : Cars are not designed to fly.

A.2 Examples of UNION model

False statement : Firetrucks are green

UNION : firetrucks are red

False statement : Plastic is more expensive than gold.

UNION : Gold is more expensive than plastic.

False statement : The sun sprays water on the planet
UNION : the sun is too hot to spray water

False statement : We need fire to survive.
UNION : Fire is not necessary to survive.

False statement : A family do not need a house.
UNION : A house is necessary for a family to live in.

False statement : Helicopters can speak English
UNION : helicopters can only fly

False statement : A normal person has 4 arms
UNION : A normal person has only two arms.

False statement : The sun comes out at night
UNION : the sun comes out at day

False statement : You should leave the lights on when you leave the room.
UNION : You should turn off the lights when you leave the room.