University of Glasgow | School of Computing Science

Contextual Query Reformulation For Multi-Domain Systems

Samuel Moyosore Kola

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

A dissertation presented in part fulfilment of the requirements of the Degree of Master of Science at The University of Glasgow

6th September 2019

Abstract

This project aims to investigate contextual query reformulation systems proposed as an alternative to co-reference resolution. Query reformulation claims to combat the various problems that plague co-reference resolution in dialog systems, such as the problem of zero anaphora and comparison anaphora. We ran various experiments using models proposed by Rastogi et al., we also used new state of the art models which they did not incorporate into in their paper. We compared the results to existing co-reference systems, transformer models, and BILSTM models with the Transformer models outperforming the others. From here, we analyzed the errors of our best-performing model and proposed a new feature, 'consistent indexing,' that could prove beneficial to query reformulation. We then extend our results by testing the effects of reformulations on document retrieval. The results were clear as the Contextual Query Reformulation systems outperformed the Irene and simple Learning to Rank systems by 179% and 31% respectively.

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name:	Samuel M. Kola	Signature:	*Samuel*

Acknowledgements

I want to thank my supervisor, Dr, Jeff Dalton who not only provided me with the much needed groundwork to kick start the project but also frequently meet up with me for discussions during his free time. I would also like to thank my close friend, Sophia Ndapandula Shilongo, who not only followed but stayed up with me at the library; this moral support can not be in any way overlooked. Finally, I would like to thank Pushpendre Rastogi for all the assistance he gave me; it was of much help.

Contents

Chapter 1

Introduction

1.1 Motivation

In recent years, there has been a surge in demand for Conversational Question Answering (CQA) agents which allow a user to query large scale knowledge bases (KB) or document collections in natural language [2]. Similarly, there has also been an increase in the demand for task-oriented conversational (dialog) systems. In this report, task-oriented conversational systems would be grouped under the umbrella of conversational QA agents because the design requires it to query databases or knowledge bases as the case may be to complete the task the user has asked it to perform. Conversational agents are being used by billions of people worldwide, they are integrated into systems, and they can be used to perform a variety of tasks from schedule setting, navigation, ordering items, playing music, and many more. The latent potential of these systems is limitless, and they can be used to do nearly any operation that a user can do on their device and can be the gateway to complete hands-free control of systems. Conversational bots and agents could have a great impact on society as it could mean physically disabled persons can more easily interact with these systems and perform everyday user tasks with no difficulty.

Accomplishing tasks in these conversational systems require the agent to not only have an understanding of context [9] but also to track turns in a dialog. A common approach used to solve spoken language understanding (SLU) is a co-reference resolution. Co-reference resolution involves resolving pronouns and other possessives in a sentence. The approach faces two major problems; the first being that it breaks down when there are no possessives or pronouns in the turns[7] and the second being that only resolving pronouns or possessives do not guarantee the query would yield relevant results if issued directly to a Knowledge Base or Search system (see figure 1.1). A better solution to this is the concept of slot carry over proposed by Chetan Naik et al. in the paper Contextual Slot Carryover for Disparate Schema[7]. The main idea in the paper is to identify relevant slots in a conversation turn then determine if to pass those relevant slots forward to the next turn. This approach, although inherently better than vanilla co-reference resolution, based on their results, has low precision and high recall in multi-domain applications. The poor results are likely because similar slots have been used for various domains[7]; they are also due to domain switching, which on its own can be difficult to detect.

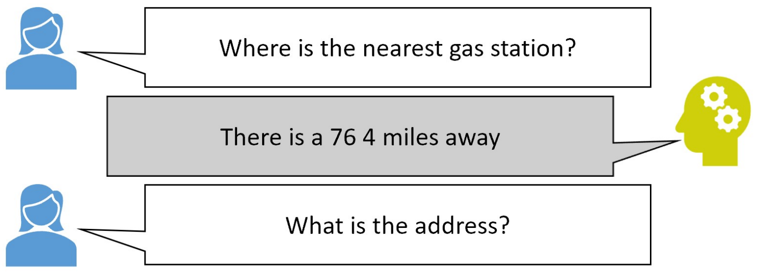


Figure 1.1: Sample dialog between a user and a system. Here the last turn of the user asks ”what is the address?” implicitly (zero anaphora) referring to the 76 gas station.

1.2 Aim and Objectives

Although work has been done on creating Contextual Query Reformulation (CQR) models [9], research into the best set of features for the query reformulation process, the best architecture for CQR, nor has the potential influence on document retrieval been investigated. This report aims to fill these gaps by investing these. The main contributions of this report include:

1. A study of the proposed models in the query reformulation paper by Rastogi et al. [9].
2. An examination of the performance of Transformers in comparison to Recurrent Neural networks for Query Reformulation
3. A proposal of an adjustment which can improve Query reformulation and co-reference
4. An investigation into the effects of CQR on document retrieval for search queries

1.3 Outline

We begin by first delving deeper into the co-reference problem and CQR, citing previous work and explaining the theoretical background of the models we implemented alongside our opinions on why we decided to try a transformer model. Here we also give a detailed description of the corpus used, including various statistics about the distribution of the documents in each corpus. Following this, we move on to discussing our implementation details. We start with a visualization of our pipeline before discussing the system requirements, the tools we used, data pre-processing, model implementation, our evaluation metrics, and the rationale behind the metrics. We then moved on to our experiments where we analyzed the results and gave reasoning to the nature of the results. Finally, we conclude with a summary of our findings, our limitations, and then possible avenues for future work.

Chapter 2

Background, Analysis, and requirements

This chapter covers the surveyed literature as well as lays the foundation on some of the key techniques and concepts used in later chapters of the report. It also describes the corpus used. We begin by first discussing our requirements which guided our selection of models and tools. Then we discuss the rationale behind our model selection. Next, we discuss the contextual carry over system. This is a simplification of reformulation as it just involves carrying over slots/entities as arguments that add more context information to a sequence. This is different from our reformulation only in that we rewrite the sequence to include the carried over slots. Finally, we introduce the corpus; here, we describe to detail the nature of the corpus how it was annotated and important statistics from the data.

2.1 Analysis and MOSCOW

The primary objective of our work is to reformulate a query using various SLU tools. To achieve this and not stray too far from our goal, we started by defining our scopes properly and identifying the requirements. Below are the MOSCOW requirement of our project

- **Must have**: As earlier stated, we aim to reformulate a query to include all resolved anaphora. This reformulation must be grammatically correct but does not need to focus on the order at which the entities appear. The system must be able to take in dialog sequence inputs and produce a single turn output
- **Should have**: The reformulations done should be able to be constructed using only tokens in previous sequences. This copy should not be the only way reformulations can be made, but having this would be a very helpful feature.
- **Could have**: The system could be used as an alternative to co-reference resolution systems as reformulations inherently resolve anaphora.
- **Wouldn't have:** The system would not perform dialog state tracking and by extension would not predict system turns. Also, the system would not predict user intent; it would only focus on carrying slots/ entities over.

With these in mind and some non-functional requirements such as; System should be easy to implement, Learning must be transferable, and architecture should be flexible, we were able to move on to our survey and identify useful techniques and tool we can employ in solving query reformulation

2.2 Transformers

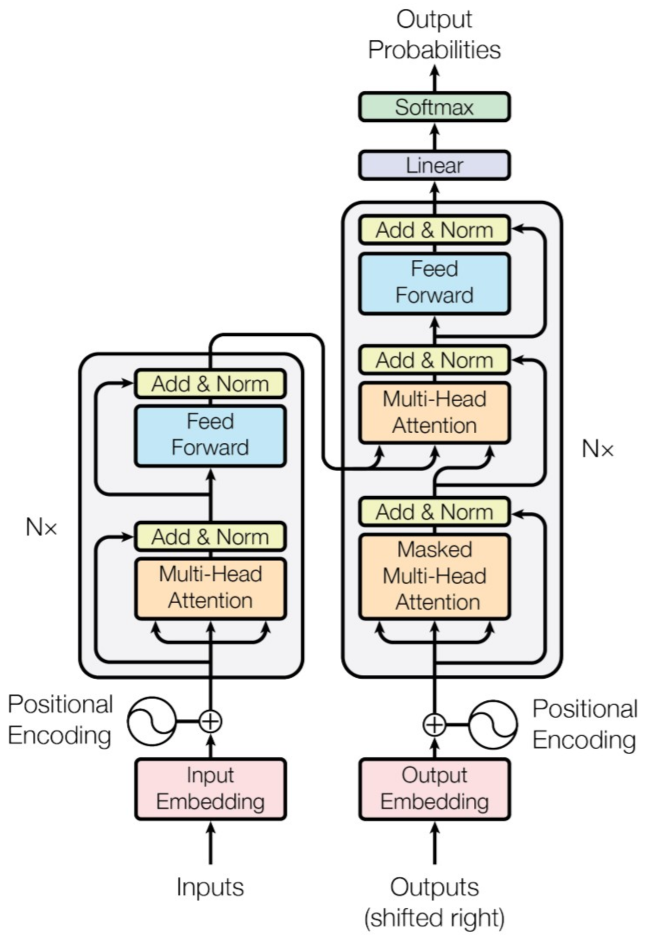


Figure 2.1: The Transformer model architecture [13]

Recurrent neural networks are primarily used for sequence to sequence translations tasks. But, research has shown that transformers could potentially produce better results [5]. Transformers also address some of the inherent problems with Recurrent neural networks. These problems are largely contributed by the RNN's architectures which make a hidden layer state dependent on the previous state and hence convert the problem to a sequence to sequence task. Although this sequence to sequence processing is desirable, it can be a potential drawback when sequence order is not very important [13]. This was the case in our reformulations, the order was not of much importance, and the reformulation could have slots positions mixed and jumbled but the meaning still the same. So a model that is not too fixated on maintaining sequence order, such as the transformer, would be preferable. Secondly, the sequence to sequence processing inherently implies that parallel processing would be difficult and complicated. Although, we did not use any parallel processing techniques since our datasets were small and using multiple GPU's might have been a waste of resources.

2.3 Pointer Generator Networks

The architecture used throughout this report for both the transformer and the BILSTM models was adapted from that of Pointer generator networks (PGN) [12]. The main reason why we used this architecture was to make reformulations more natural and not template-based. This would allow us to reduce the focus on the grammatical structure, which is not the problem we are trying to solve. The PGN's can produce such natural-looking sequences by utilizing a vocabulary set which the authors of the PGN paper termed the attention distribution.

They also incorporated a generation probability into their model, which made it possible for new sequences to be generated either by repeating token in previous sequences or by selecting from the distribution available. These newly generated words are often contextually correct as they based on the embedding vectors which have similar words having similar embeddings. The ability to copy from previous sequences is also quite essential as reformulations most often have very similar words. All these features embedded into the architecture of the PGN make our model more flexible and allows it to focus on the task we have set for it more easily.

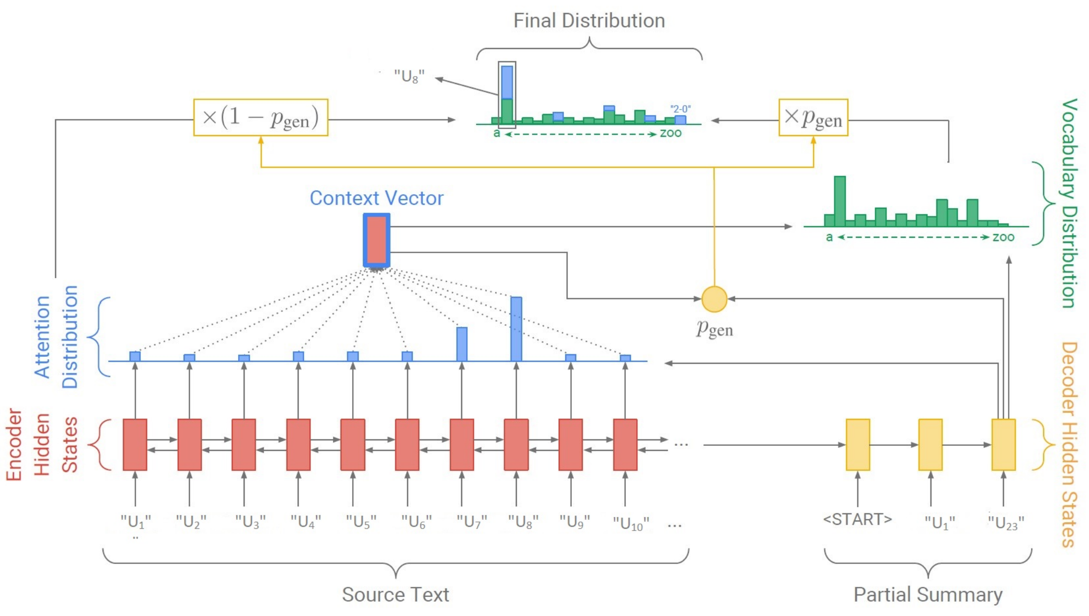


Figure 2.2: Pointer generator network architecture, [12]

2.4 Contextual Slot Carry-over

Slot filling is the common paradigm in dialog systems, with users being able to refer to slots in the previous turn. We investigate the work of Chetan Naik et al. [7]. The main idea of their work is solving the resolution problem by phrasing it as a slot carryover where they assign a probability that a slot would be needed or is being referred to in the current turn. This is particularly useful as it makes their model somewhat independent of linguistic features [7] which are not necessarily important in solving the co-reference resolution problem. This means that zero anaphora problems, which is a hard problem for co-reference resolution, would be dealt with in a similar way to nominal, and pronominal anaphora problems. Similar to this is the frame tracking model [11]. Here, entire user goals are tracked, and the main aim is to allow users to switch back and forth between domains and tasks.

Task definition and model architecture

The task for the contextual slot carryover is to predict the probability of a slot or set of slots S_t to be carried over given a dialog act a_t and sequence of tokens w_t at any given time t. A slot is carried over if the probability is greater than an optimized threshold, τ. See the figure 2.3 for the model architecture .

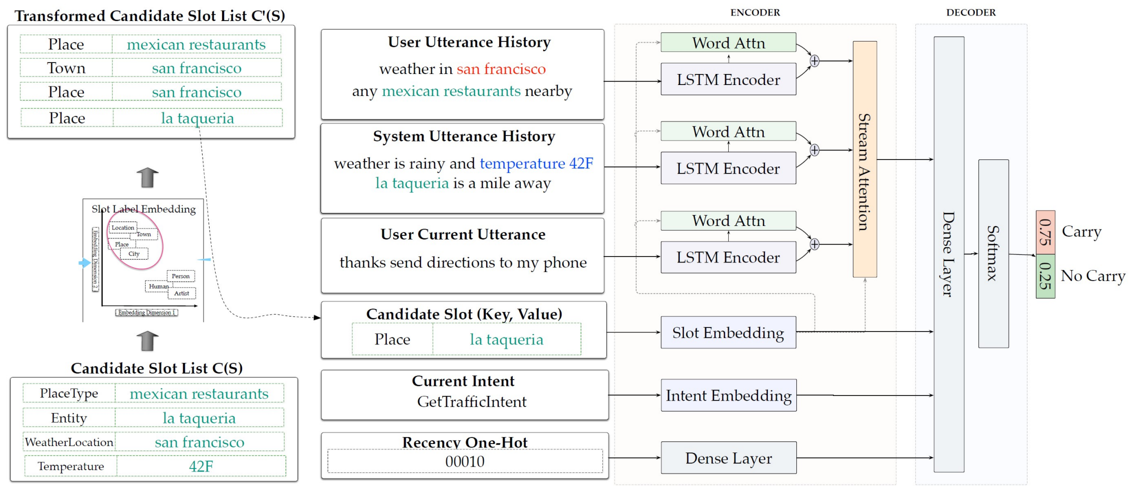


Figure 2.3: The architecture of the encoder and decoder layers for the contextual slot carry over model [7]

2.5 The Corpus

This section of the report describes in detail essential statistics of datasets that were used and included but is not limited to information about how the corpora we used was developed, useful information about the distribution of data and problems with the data.

We used two main corpora for our analysis, the first was the Alexa dataset contextual query rewrite data which was used to train and develop a state of the art query re-writer using transformers and recurrent neural networks and the second was the Treccast data which we used to not only test our hypothesis of query rewriting improving the performance of conversational question answering but also to train a network to perform multi-domain query rewriting better .

2.5.1 Alexa data-set for contextual query rewrite

The Alexa dataset [1] contextual query rewrite is an extension of the publicly available Stanford dialog corpus [1]. The Alexa dataset for contextual query rewrite aims to re-model the co-reference resolution problem as a query reformulation task where slots would be carried over, and it particularly targets solving the problem of zero anaphora [10]. The process of rewriting ambiguous queries in the multi-turn dialog is done using a set of guidelines which we summarized from their paper as:

- Extract the phrase which most completely conveys the user's intent or request, i.e., the basis utterance
- Reformulate the basis utterance making all the user's intent unambiguous.
- When a location is not explicitly being referred then assume the place to have an implicit reference then resolve.
- Not all turns require reformulation, e.g., confirmation and gratitude
- Slots from context replace an anaphoric reference
- Only anaphora related to the domain of the turn being reformulated need to be handled
- When multiple values for a slot are present use most specific slot
- Solve utterances with multiple anaphora to the same slot
- User intent can be carried over where needed

Please read their paper for a more detailed break down of their guidelines.

Descriptive Statistics

The Alexa Data set contains dialogs categorized into Navigation, Schedule, or Weather. It has 2131 dialogs in its training set, 271 in the development and 276 in its development set. About 85% of this Data had reformulations done on them using the above guidelines. Figure 2.1 shows a more detailed description of the types and frequency of anaphora in the corpus.

Dataset	Reformulations	Zero	Pronominal	Locative	Nominal
DEV	206	143	31	34	21
TEST	214	155	21	43	20
TRAIN	1867	1138	393	162	143
Total	2287	1436	445	239	184

Table 2.1: The distribution of anaphora in the Alexa dataset (zero, pronominal, locative, nominal) [10]. The majority of the anaphora is implicit, i.e zero anaphora with no nouns or pronouns to resolve..

The corpus had 15 distinct slot types, the frequency of 13 of them can be found in Table 2.2, below. We excluded two of these slot types from this table, and they were the 'agendas' and 'day'

[1]https://github.com/alexa/alexa-dataset-contextual-query-rewrite

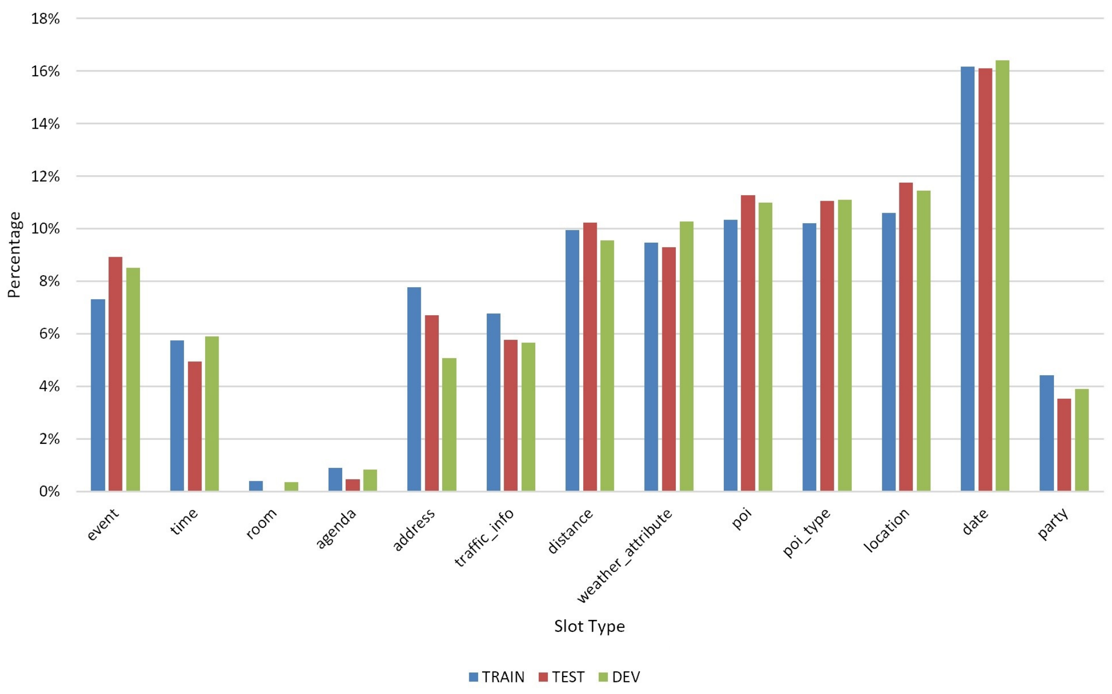


Figure 2.4: Slot Type distribution in the Alexa dataset

slot types. These were lumped under agenda and date, respectively. The reason why we did this was that they both had a frequency of one, and only appeared in the training data.

Name	date	party	room	weather_attribute	poi_type	location	address
TRAIN	1201	329	29	704	758	787	577
TEST	137	30	0	79	94	100	57
DEV	139	33	3	87	94	97	43

Name	traffic_info	time	event	agenda	poi	distance
TRAIN	503	429	544	66	768	739
TEST	49	42	76	4	96	87
DEV	48	50	72	7	93	81

Table 2.2: Frequency of all 13 slot types in the Alexa dataset.

Nearly all slot types were evenly distributed across the training, development and test datasets.The outliers being date; which was used frequently across all categories (Weather, Navigation and Schedule), agenda; which was only present in scheduling tasks and room; which was also only present in a few scheduling task (see Figure 2.4).

2.5.2 TREC CAsT Data

The TREC CAsT data [2] is also another publicly available dataset aimed at providing data for training and evaluation of conversational question answering agents. The data was created to chal-

[2]https://github.com/daltonj/treccastweb/tree/master/2019/data

lenge the current co-reference resolution systems by providing data with hard and complex resolutions. The dataset did not contain gold reformulations like the Alexa data provided by amazon, so we manually rewrote all the queries using a modification to the guidelines [10] used to rewrite the Stanford dataset into the Alexa data set. These alterations to the rules were mainly because the Data was not in the same format and also did not have slots. The changes we made to the guidelines include:

- Resolving comparisons to include both slots being referred. E.g. ”Which is better?” Would be resolved to ”Which is better A or B?”; where A and B are the entities being referred too.
- Expanded domain-specific ”abbreviations,” we also included incomplete entity phrases as abbreviations and expanded them, e.g., turn 1: what are chemical reactions. turn 2: tell me more about reactions; the reactions here would be expanded to chemical reactions.
- Where reference was difficult to determine, resolve to closest entity mention
- Entity tense and quantity are maintained in rewrites. We decided not to focus on ensuring that the reformulated queries have the entity in the right tense but rather focused on carrying over the entities as they were. This was done to simplify the string matching done when pre-processing

Descriptive Statistics

The Treccast data consisted of 30 dialogs in the Training set and 50 in the evaluation, each containing only user turns. Each dialog in the dataset had about nine turns each, with about 71% of these turns containing at least one reformulation using the above rules. The reformulations were categorized into; nominal for noun resolutions, pronominal for pronouns, zero for zero anaphora, groups for comparison words which required two entities and finally abbreviations which contained expanded abbreviations.

Name	Nominal	Pronominal	Zero	Groups	Abbreviations
TRAIN	5	97	82	6	29
TEST	4	122	111	4	15

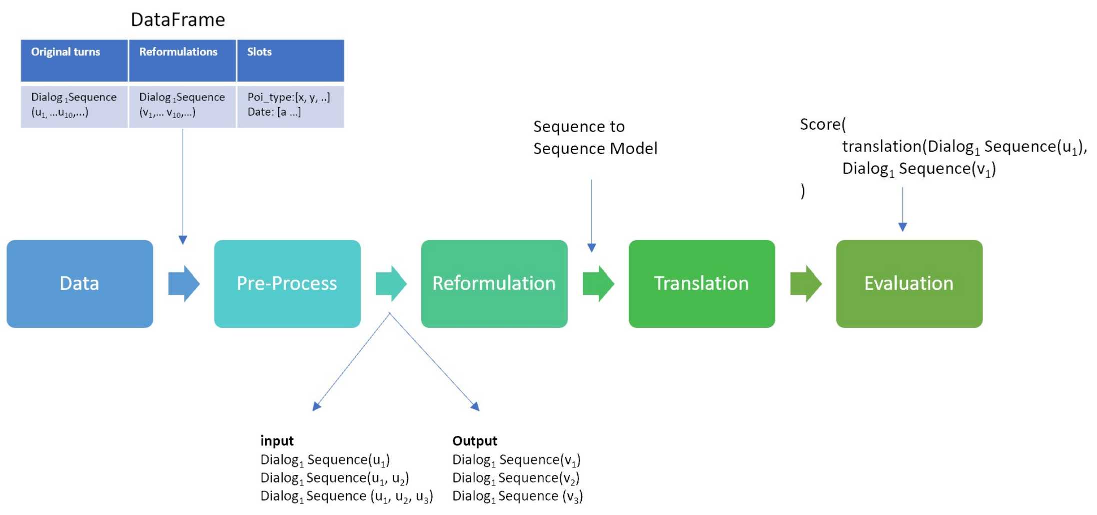
Table 2.3: Distribution of anaphora (Nominal, Pronominal, Zero, Groups and Abbreviations) in TREC CAsT Dataset.

Chapter 3

Design and Implementation Details

This chapter builds on the theoretical groundwork in the previous and discusses in detail how the models and algorithms can be implemented. We start by showing a visualization of our process and briefly describe the tools and infrastructure used. Following the order in the pipeline, we first describe the prepossessing, the inputs to the model, then move on to focus on the main tool OpenNMT [4]. Here we describe how the Recurrent Neural Network and the Transformer models were implemented. We then round up by discussing our evaluation metrics.

3.1 Pipeline, Tools and Infrastructure



DataFrame
Original turns
Reformulations
Slots
Dialog 1Sequence (u1, …u10,…)
Dialog 1Sequence (v1,… v10,…)
Poi_type:[x, y, ..] Date: [a …]
Sequence to Sequence Model
Score(translation(Dialog1 Sequence(u1), Dialog1 Sequence(v1))
Data
Pre-Process
Reformulation
Translation
Evaluation
input
Dialog1 Sequence(u1)
Dialog1 Sequence(u1, u2)
Dialog1 Sequence (u1, u2, u3)
Output
Dialog1 Sequence(v1)
Dialog1 Sequence(v2)
Dialog1 Sequence (v3)

Figure 3.1: Data Pipeline for our experiments

Using the MOSCOW requirements and objectives of our project, we constructed a pipeline that we used to measure progress. To conduct our experiments successfully, we have leveraged various open-source tools (mostly libraries). We leveraged commonly used data science tools used in deep learning and text processing for our pre-processing such as:

- spacy 2.0.18,
- pandas 0.24.2, and
- numpy 1.16.2

We also decided to use the open-source toolkit for neural machine translation (OpenNMT) [4] primarily due to its ease of implementation. The toolkit is also a commonly used toolkit for machine translation which would make our work easily reproduced by other researchers.

Infrastructure

Since the datasets used to train our models were not large, it was possible for us to perform all pre-processing locally on a 16GB CPU, with 4 cores. With these and optimization's to our code, we were able to perform pre-processing tasks comfortably. We used the same CPU specifications for our evaluations. For remote access and monitoring, we trained and tested all our models using Google Colab's provided environment. The environment was run on Python 3 and 12GB GPU, which proved very useful in speeding up the training time of our models. Similarly, we stored our data files on Github and accessed them by cloning the repositories and storing them in a folder on Colab.

3.2 Processing the Data

This section discusses how we used the text preprocessing tools described above to add features to our data. It also describes the input and output format shown in figure3.1. The section then concludes with how we stored the extracted attributes from the corpus to increase the efficiency of our runs.

3.2.1 Pre-processing the Alexa Dataset

This section describes how we carried out the pre-processing. The main goal of this section is to give much detail about the inputs and outputs format used in training the sequence to sequence models. We employed python for all pre-processing, and nearly all code used was written line by line from scratch as the processing code used in the Rastogi et al. [9] paper, was not open-sourced.

Entity Indexing

We used two kinds of Entity Indexing in our pre-processing, one was the Role-Based Entity indexing we were replaced entities with their indexed slot types [9], in the other we replaced entities with the indexed string 'entity,' i.e. $entity\_i$ where i is an index for the set of slots seen in the dialog.

The former was done to improve the performance of the model by making it just learn how to carry over slot types rather than learning to both identify then carry them over. The latter indexing style was used as an alternative to the role-based entity when the data was being used to pre-train a model for use on the treccast Data for conversational question answering. This style was adopted as the treccast data had not slot types and using a network pre-trained on slot types then fine-tuned with the treccast data gave poor results. About 9% of slots in the dataset were not indexed. The reason for this un-indexing was because, during the reformulation process some of the entity's names were changed to better descriptors, but since the names did not match the entities in the slots, they were not matched and hence were unindexed.

Syntax and other features

It is commonplace to believe that the addition of syntax-based features can improve the performance of an SLU system. This follows the intuition that the syntax and other lexical features convey more information about the context in a sentence [3]. We have thus included syntax features as input to our model. We attempted to replicate similar features added to the model in the CQR paper by Rastogi et al. [9] as closely as possible. We similarly appended QUESTION to the 9 (not seven as indicated used in the paper) question words, appended PSBL to possessives identified by the spacy dependency tagger and finally we appended PRB$ to pronouns which were not already previously classified as possessives by the dependency tagger.

I/O format

Conventional input and output format sequence to sequence models used in dialog state tracking tasks were employed. The input to the system was a dialog between the user and the system, and the output was the reformulated gold queries of the last user turn not the system response. All dialogs used had multiple turns so dialogs were ultimately turned into $N_i/2$ I/O pairs, where N is the number of turns in a given Dialog i. The dataset has a median of 6 turns $(N = 6)$ which means the total number of sequences to train, develop and test on would increase from 2131, 271, 276 to 6019, 781, 744 respectively.

Pre-processing the TREC CAsT dataset

The pre-processing done on the Treccast dataset was similar to that done on the Alexa dataset. The I/O format was nearly identical except for the missing system turns. This also meant the sequence input length increased by N not $N/2$. The Treccast data also did not contain slots, so role-based entity recognition was not performed on it. Although we initially planned to use wikification to label slots, we quickly abandoned the idea as our pre-trained model on Alexa data would not have enough data to learn all the new slots that would be obtained. So we decided to use only and indexed count where we replaced potential slot with $entity\_i$ as done similarly with the Alexa data. No added features were added to the Alexa data following results gotten from the addition of other features to the Alexa data.

3.2.2 Efficiency in processing

The pre-processing steps described above involves a lot of string matching, tokenization, sentence scanning, and many other string operations which nearly all have a complexity of $O(n)$. To avoid having to do all this processing, we have stored the results of operations that could potentially slow down the process. We stored our results in a pandas data frame which we converted to a JSON file. Two main data frames were derived from the Alexa data. The first being a table with column names; $Original\_turns$, $Original\_Role\_Based$, $Reformulated$, $Reformulated\_Role\_Based$ and $Slots$. The columns with the suffix Role_Based were role-based entity indexed versions. The second data frame was a table used to speed up the evaluation process. It had columns with names, 0 to 3 whose row elements contained a set of all the entities that can be gotten from those distances, with column 3 containing all entities gotten from 3 and above.

3.3 OpenNMT

Opensource toolkit for Neural Machine Translation, OpenNMT [1], is a deep learning toolkit used for training, developing, and testing models aimed at solving Spoken Language Understanding problems. Havard developed OpenNMT as a system to replace $seq2seq - attn$ which performed similar functions. OpenNMT is very flexible and allows users to adjust parameters without worrying too much about the implementation details. The toolkit was developed on GitHub using python (http://github.com/opennmt/opennmt) and is primarily maintained by the Harvard NLP group and SYSTRAN Paris.

We first begin by introducing OpenNMT's pre-process, which is used to create the word embeddings for the training and development data. It also includes an option that allows the use of external pre-trained embeddings such as the GloVe or Bert embedding. Some of the important arguments that can be passed to this are:

- *share_vocab* – This option allows us to decide if to share vocabulary between the inputs (source) and the outputs(target). It means that when creating word embedding, it would use both sequence information from the source and target. It is particularly useful in addressing the problem of are words in small datasets. In other words, it reduces the chances of having $< unk >$ tokens when testing or developing the model. We set this for all the models we trained as both data sets we use were relatively small.
- *dynamic_dict* – The dynamic dict creates a python dictionary for all sentences in the inputs of all the tokens in the sentence. This option is particularly useful when an attention copy mechanism has been implemented.

The share_vocab option was the main option we changed in all the models, and all other features, unless otherwise stated, were let at their default values.

Next is OpenNMT's train, this as the name implies deals with the necessary parameters needed to train and develop the model. The main arguments to this are the hyper-parameters of the model and the type of neural network used to solve the sequence to sequence tasks. A few useful parameters are:

[1] `https://github.com/OpenNMT/OpenNMT-py`

- *train_from* – This primarily allows training to continue from a saved checkpoint. It is particularly useful in transfer learning. Saved models trained on different data can be loaded and fine-tuned. This feature was particularly useful when dealing with small datasets. Which meant complex features and transformations could be learned using larger datasets and fine-tuned using the smaller data set.

- *word_vec_size* – the word vector size option is used to set the size of the word embedding for both source and target. Alternatively, they can be set individually for source and target. When setting this parameter, we made efforts to set it to the smallest possible size that would not compromise performance. We used smaller embedding sizes so models would be forced to learn denser representations of the words in the source and target.

- *encoder_type and decoder_type* – We used this to set our model to be either a recurrent neural network or a transformer. The toolkit also has options for convolutional neural networks. For translation and sequence to sequence tasks, RNNS and transformers have proved better suited, so we decided only to explore them.

- *rnn_type* – If either both or one of the encoder and decoder types are RNN then this option allows to select the type of recurrent neural network used. The default is the BILSTM, so we left it at the default value, but the toolkit also supports the use of Gated Recurrent Units and Simple Recurrent Units

We trained the BILSTM model (see appendix for model architecture) using 10000 steps, fewer steps sizes such as 5000, 1000 and 100 were used at the initial stages, but we observed that the models did not converge so we increased the epoch size till we observed convergence. We also used an embedding size of 128 for dense representations, a learning rate of 0.15 and validated every 100 steps, as shown below:

```
#@title RNN model
!python -u ./train.py -data preprocessed -save_model CQR_valid_100\
        -train_steps 10000 -rnn_size 128 -word_vec_size 128 \
        -encoder_type rnn -decoder_type rnn -optim adagrad \
        -learning_rate 0.15 -share_embeddings -valid_steps 100 \
        -save_checkpoint_steps 1000 -log_file log.txt
```

For the transformer models, we needed to set a few more parameters such as the number of layers which we set to 6, the number of attention heads which we set to 8, dropout which we set to 0.1 and the batch normalization. Other features we set to be valued as closely matched to the BILSTM as possible to allow fair comparisons between model performance. Below is the code snippet containing all the parameters we used for training our transformer models:

```
#@title Transformer model
!python  train.py -data preprocessed -save_model CQR_valid_100 -layers 6\
        -train_steps 10000 -valid_steps 100 -transformer_ff 2048 -heads 8\
        -src_word_vec_size 512 -tgt_word_vec_size 512 -rnn_size 512\
        -position_encoding -encoder_type transformer -decoder_type transformer\
        -max_generator_batches 2 -optim adam -warmup_steps 800 -dropout 0.1\
        -batch_size 256 -batch_type tokens -normalization tokens -accum_count 2\
        -adam_beta2 0.998 -decay_method noam -max_grad_norm 0\
        -learning_rate 0.15 -param_init 0  -param_init_glorot\
        -label_smoothing 0.1 -save_checkpoint_steps 1000 -log_file log.txt
```

Finally, OpenNMT has a translate option which can be used to test the model. It simply applies the learned transformations to source sequences. Similarly, we used default parameters for this, but it has some useful parameters such as"

- *dynamic_dict*– this is useful indicating if to use the attention copy during the translations or not. The dynamic dictionary was no used unless specified.
- *report_bleu*– this calls a multi-bleu tool that calculates the bleu score of the translation. We used this option when we needed to calculate the bleu score of our translations.
- *beam_size* – sets the maximum number of states stored at each level when creating the search tree.

3.4 Evaluation metrics

Entity F1 We measure the macro averaged F1 score at various turn distances, d, here. We did this to be able to see how the model performs in resolving anaphora that is varying distances away from the current turn. We calculated the macro F1 at $d = 0, 1, 2, 3+$. We stopped at $d >= 3$ because we observed that most anaphora that needed resolving were within the 0-3 margin and only a few were more than three turns away. Our calculations of precision and recall also slightly different from those used in the Rastogi et al. [9] paper, where they used the micro F1 score. We decided on this different way of estimating these values as their values were too optimistic and did not equally weigh all predictions. We included the number of overall wrong slots/entities in the calculation of the precision and recall for all distances, were as they did not. This made our values slightly less, but since we used the same method to calculate the F1 on all our models, we could still compare across models.

BLEU The BLEU score, proposed by Papineni et al. [8], is commonly employed in metric for grading translation tasks. It aims to mimic the performance of human evaluators. It does this by calculating the similarity between the predicted text and the actual text on an n-gram based level. This allows it to be able to capture both semantic and syntactic meaning by simply calculating the similarity with different values of n. The main weakness of the scoring is it can give low scores to word and sentence variations. Although we used this scoring metric to check how closely the model emulated the actual text, it is not the best measure of performance for our results as we are concerned with the slots carried over and anaphora resolved.

ResF1 This measures the micro F1 score across domains. We used this to gauge which domains were harder or simpler than others. It allows us to capture more domain-specific improvements when testing models and also allows for more refined error analysis. The precision and recall used to calculate this was similar to that which we used in the Entity F1 score. We use this when we propose a new feature. It allowed us to show the domain-specific drawbacks and improvements of the feature.

MAP Mean Average Precision is a widely used metric to measure retrieval effectiveness. We use this only when demonstrating the possible effects the CQR model can have on document retrieval.

Chapter 4

Experiments and Evaluation

In this Chapter, the experiments, evaluation, and results are discussed. The experiments and evaluations can be split into two distinct groups. The CQR group and the TREC CAsT group. The CQR group focuses on how closely the model is able to match the golden reformulations while the Treccast group focuses on both the match to the golden reformulations and the improvement in results when the reformulated queries are used to retrieve documents. The CQR group only focuses on the Alexa data set while the TREC CAsT group focuses on extending the results from the Alexa data to a document retrieval problem using the Treccast data.

4.1 CQR Setup

We decided to use 3 baselines to compare the results of our model best. We used two existing co-reference tools, spacy's neuralcoref by hugging face and AllenNLP by Matt Gardner et all [2]. We also decided to include a simple copy of the turn being reformulated so we can see how much better or worse our model is than simply doing nothing.

4.1.1 Implementation detials

NeuralCoref and AllenNLP were used out of the box, and there was no further training done on them. All other models we used were trained and validated using the Alexa TRAIN and DEV dataset. We trained the CQR model using a BILSTM network as done in the Rastogi et al. paper [9] for an epoch size of 100. This gave very poor results as the model had not converged, so we decided to train on a larger epoch size, 10000. This produced better results but took an average of 3 hours to train. We also encountered two local minima at the validation accuracy of 11.5744 and 11.5798. If the validation accuracy reached these values while training the model would cease to improve, but go back and forth to smaller values then back to it. This didn't occur often, but when it happened, all we did was retrain the model and let the random initialization solve this problem for us. The transformer models were also trained in similar ways to the BILSTM model. They took about 10 hours to train. We saved the models every 1000 steps and used graphs of the validation accuracy vs. the steps and perplexity vs. step to determine the best model. We often selected the model saved at step 6000 or 7000. We agreed that models saved at steps higher than this would overfit while those less would underfit.

We also performed experiments with using pre-trained embedding such as the GLoVE embedding, but the results were the same as when we did not. Although, it is important to note that the model with the pre-trained GLoVE embedding initially started of doing better in the validation set, but later converged to almost the exact same results for both the accuracy and the perplexity. For this reason, we decided to exclude it in the testing but included the validation results in figure 4.1.

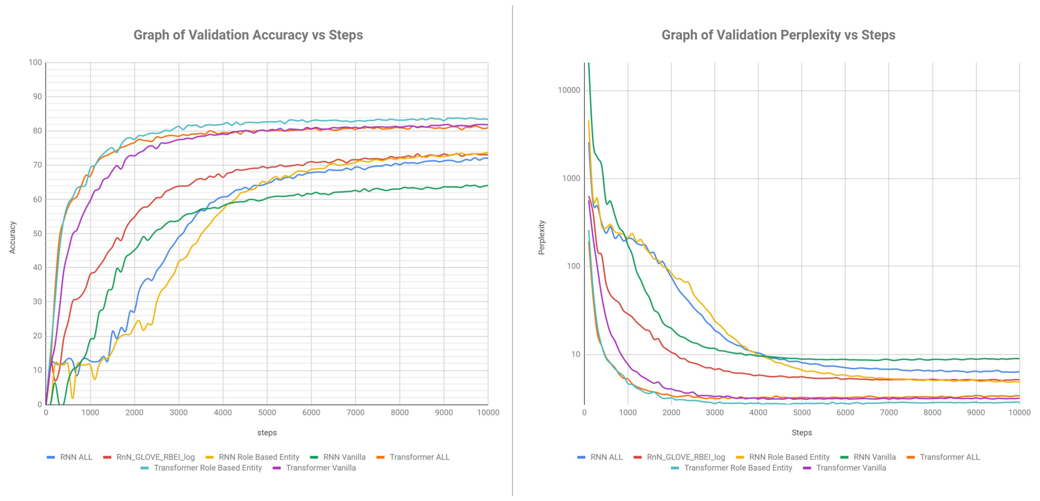


Figure 4.1: Perplexity accuracy vs steps

Overall the models followed similar curvatures and maintained their order with the transformer model outperforming the BILSTM models consistently in both accuracy and validation.

4.1.2 CQR Testing and Evaluation

System	Entity F1												BLEU
	d=0			d=1			d=2			d >=3			
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	
AllenNLP	0.935	0.926	0.929	0.194	0.132	0.151	0.193	0.152	0.164	0.411	0.199	0.257	54.070
Copy turn	0.966	0.965	0.966	0.127	0.086	0.099	0.176	0.141	0.149	0.363	0.177	0.227	58.990
Neuralcoref	0.957	0.954	0.955	0.215	0.145	0.166	0.199	0.170	0.176	0.379	0.187	0.240	57.410
RNN Vanilla	0.304	0.234	0.254	0.148	0.090	0.106	0.176	0.155	0.162	0.199	0.108	0.130	18.660
+ Role Based Entity	0.718	0.780	0.729	0.580	0.772	0.630	0.661	0.868	0.699	0.553	0.785	0.626	37.850
+ Syntax Info	0.696	0.838	0.736	0.531	0.753	0.593	0.610	0.918	0.694	0.482	0.703	0.552	28.290
Transformer Vanilla	0.814	0.774	0.781	0.672	0.572	0.593	0.744	0.771	0.743	0.649	0.575	0.581	47.520
+ Role Based Entity	0.901	0.937	0.910	0.804	0.896	0.830	0.851	0.972	0.886	0.716	0.853	0.763	60.740
+ Syntax Info	0.889	0.912	0.893	0.743	0.833	0.765	0.846	0.939	0.875	0.557	0.678	0.594	63.740

Table 4.1: Comparison of Precision, Recall, and F1 scores at different distances from the reformulated turn. Our Transformer Role-based Entity model beats the BILSTM models on all metrics

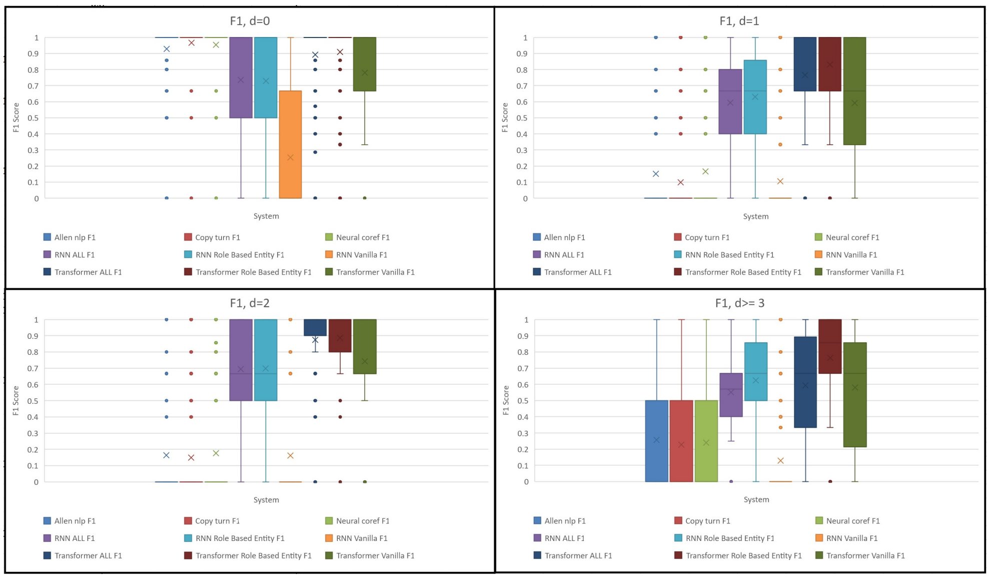


Figure 4.2: Box and Whiskers of all models showing the mean (X), median and quartile ranges of the F1 scores at various values of d. It can also be noticed that the problem gets much harder from $d >= 3$ as models begin to have mean values of 0

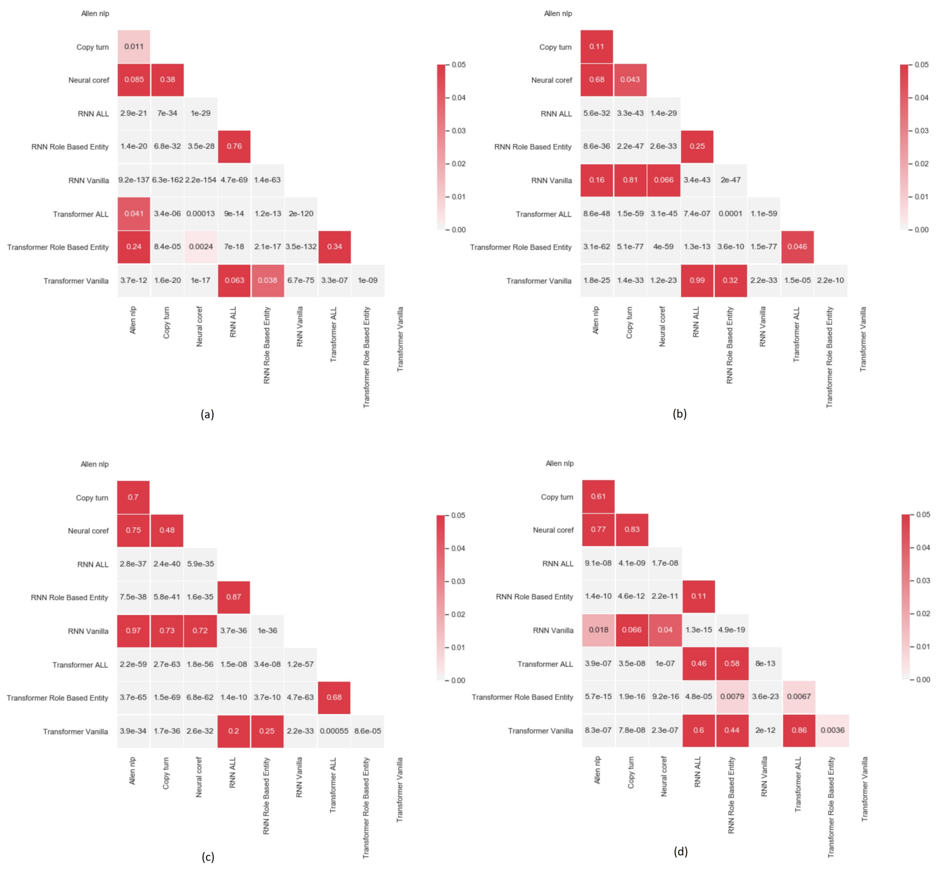


Figure 4.3: Confusion matrix of p values from a T-test comparing the differences between the results of all models. We only accepted the results as different if the $p < 0.05$

Using precision, recall, F1, and BLEU, we evaluated the performance of our models and compared them to baselines discussed in the previous section. Our results confirmed our hypothesis that the Transformers would perform better in sequence to sequence tasks than RNN based models and supports the results in Surafel M. Lakew et al. [6], where they compare the performances of RNN networks to Transformers.

We also looked at the spread of the results in order to determine how consistent the results of our models were. We did this by plotting boxplots of the F1 scores at various values of d, Figure 4.2. From here, we were able to confirm that the Transformer Role-Based Entity model was indeed the best performing model with it consistently having a majority of its F1 score in smaller ranges. We also noticed that the RNN model with all features although performed worse than the RNN with only Role-Based entity indexing, still was a more consistent model which would make it more reliable to use in practice.

Finally, we decided to investigate the difference between the results of our models by performing two-tailed T-tests. We did this because the difference between the evaluation scores between some of the models were < 0.5. We thus decided to determine which models were most likely to produce similar results when used in practice. We observed that from $d >= 1$ the RNN vanilla model, Neuralcoref and AllenNLP were just as good as doing nothing to the query. We also made the interesting observation that the Vanilla Transformer model produces similar results to that of the best performing RNN models. This further backed our claims that Transformer models outperform RNN models, even without added features.

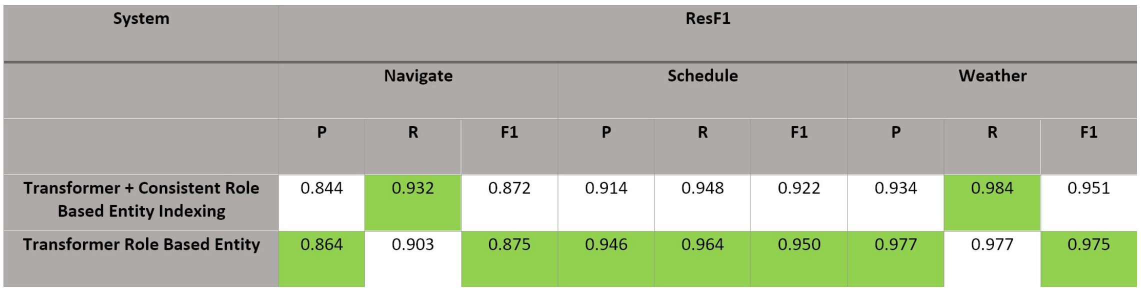
4.1.3 Analysis

After establishing our Transformer model as being superior, we began decided to examine the results of our best performing model, the Transformer + Role-Based Entity Indexing (RBEI) with the aim of identifying errors and developing new features that could potentially improve performance. We noticed that the Transformer RBEI model was not completely missing the entities. It was most often getting the correct slot type but was only giving the wrong indexes to the slot types, e.g., poi_1 instead of poi_2, see table 4.2. We thus proposed a new feature where we maintain index observed in each sequence by relabelling and role-based entities. We called this labeling consistent role-based entity indexing.

After re-indexing the role-based entities using the consistent index we decided to compare across domains using the ResF1 metric. We did this because we observed that the error was more prominent in the navigation and weather domains. We compared our results to those of the transformer model with the simple role-based entity indexing. Our results are summarized in table 4.3.

	Sequence (dialog)	Slots
Gold	<**user**> give me directions to nearest mall <**system**> There is Ravenswood Shopping Center 5 miles away from here but with heavy traffic on our way <**user**> Give me the address of the mall Ravenswood Shopping Center 5 miles away	**distance**:(nearest, 5 miles away) **poi_type**:(mall) **poi**:(ravenswood shopping center) **address**:(434 arastradero rd)
Role Based Entity index	<**user**> give me directions to distance_1 poi_type_1. <**system**> there is poi_1 distance_2 from here but with heavy traffic on our way <**user**> Give me the address of the poi_type_1 poi_1 distance_2	
Consistent Role Based index	<**user**> give me directions to distance_2 poi_type_1. <**system**> there is poi_1 distance_1 from here but with heavy traffic on our way <**user**> Give me the address of the poi_type_1 poi_1 distance_1	

Table 4.2: Example showing Role based entity indexing vs Consistent Role Based entity indexing. The consistent Role based indexing aims to have the all last turns of the user in a similar index. In other words, the indexing would always start with the turns as index 1



System	ResF1								
	Navigate			Schedule			Weather		
	P	R	F1	P	R	F1	P	R	F1
Transformer + Consistent Role Based Entity Indexing	0.844	0.932	0.872	0.914	0.948	0.922	0.934	0.984	0.951
Transformer Role Based Entity	0.864	0.903	0.875	0.946	0.964	0.950	0.977	0.977	0.975

Table 4.3: Table comparing the precision, recall, and F1 scores across domains for both the Transformer with Consistent role-based entity indexing and that with the normal role base entity indexing. The results overall performance of the two models is similar but with the consistent indexing giving higher recall values in Navigation and Weather

The results show that the order in which the slots are indexed affect the performance of the model, this was particularly useful for tasks that had more than one mention of particular entities (such as the navigation and weather tasks). Since recall is more important than precision for queries involved in document retrieval or database query because the presence or absence of keywords can influence the performance, it's safe to conclude that the improved feature improved the effectiveness of our CQR model. It is also important to note that the model trained about 5 to 10% faster and had narrower ranges for both navigate and weather domains.

4.2 TREC CAsT results

The aim of co-reference systems or slot carryover systems is ultimately to improve document retrieval. Following the promising results we obtained with the query reformulation, we then investigated the effect on document retrieval.

4.2.1 Preliminary analysis

Although the results of the CQR model were promising we decided to test the performance assuming the model was able to make perfect rewrites, which is of course not completely possible but should serve as proof of concept. We ran the reformulated queries against the Irene baseline and a simple LTR model. We were able to get a MAP score of 0.340 which was 31% better than the LTR model and 179% better than the Irene baseline model in retrieving useful documents, see figure 4.4.

Metric	Irene Baseline	Simple LTR	Golden rewrites
MAP	0.122	0.259	0.340

Table 4.4: MAP scores of different systems. The golden rewrites does significantly better than the other models

4.2.2 Pre-training the Model

Having established that the CQR model would perform better on the document retrieval, we decided to move on to training a model that would perform reformulations on the TREC CAsT data. The idea was that if we can achieve an overall recall of $>= 0.80$ the model would be able to outperform the existing systems in document retrieval. Since the TREC CAsT dataset was a small one. We decided to pre-train using the Alexa dataset, with the best performing model, then using transfer learning fine-tune the model on the Treccast data.

The Alexa data had slots, and slot types which made it easy to perform the role-based indexing used in the best performing model (i.e., the transformer model) but the Treccast data, has established in the previous chapter has no such features. So we used entity-based indexing discussed in chapter 3 to label the Alexa data for the pre-training and also used it to label the Treccast data. As a baseline, we decide to compare the results to that of spacy's neuralcoref. We used neuralcoref as our baseline because the anaphora is primarily pronominal, which should be advantageous to neuralcoref. We divided the training data to a 60 - 40 split of training and validation respectively. We split the already small training data because when we were fine-tuning the model, the evaluation data was not yet made publicly available. It only became available close to completion of the project and so was not involved in the training nor evaluations.

System	**ResF1**		
	P	R	F1
NeuralCoref	0.267	0.433	0.317
Transformer model	0.351	0.532	0.403
+ Role Based Entity Indexing	0.429	0.664	0.496

Table 4.5: Table comparing the precision, recall and F1 scores for Transformers vs current best performing open source co-reference resolution system we discovered (neuralcoref)

Although the training and testing data were both small, we were able to show that the proposed transformer CQR model performs much better than state of the art co-reference resolution. Given, more data, we are confident that the model would easily learn proper transformations that enable it to reformulate queries and be a very useful feature in document retrieval.

Chapter 5

Discussion

5.1 Conclusions

The primary aim of the paper was to propose conversational query reformulation as an alternative to co-reference resolution. In doing so, we undertook many experiments. We investigated the results of the conversational query reformulation system proposed by Rastogi et al. There we found that query rewriting indeed performs much better than co-reference resolution and is more able to resolve cases with zero anaphora; which is very difficult. This propelled us to perform more experiments and propose one new model architecture and a new possible feature. The proposed transformer architecture was able to improve on the already well-performing RNN (BISLTM), with little modification to the default hyperparameters. These results imply that with further experimentation very accurate results could be achieved. We believe that recall in the high 90% range is possible given properly selected features as our transformer-based systems were not getting reformulations completely wrong. The systems were mostly mixing up entity types that were very similar to one another, e.g., if there are two locative mentions, it might fail and select another location to be carried over. This lead to our development of a new useful feature called consistent indexing which proved to be a useful feature in improving not only the recall in such cases (weather and navigation) but also the time to overall model convergence and the spread of the results in such specific cases. This feature also had its drawbacks; it worsened the results in the scheduling domain but not very significantly. The effect of this worsening is reduced by applying consistent indexing only to the domains that have higher propensities for multiple similar entities (aka multiple items on the same slot type). As earlier said, we believe that further investigation into query reformulation as an alternative to co-reference resolution can lead to better features and very precise, and accurate systems.

5.2 Limitations and Future work

One thing we could not do was modify the learning objective as done in the Rastogi paper [9]. Numerous efforts were made to implement it, but their instructions were not clear enough and only explained on a surface level what needed to be done. In reality, adjusting the loss function of the OpenNMT toolkit required much more tweaking than they had let on. We contacted the main author, Pushpendre Rastogi, and relayed our progress but he could not be of much help due to company restrictions but admitted to them having to do a lot of work before being able to implement their Multi-Task Learning objective function. The results of the project were according to him used to improve Alexa, amazon's Spoken language understanding system. This, of course, did not affect the quality of our results, nor does it refute any of our findings. The transformer models we used did not have their objective function tweaked, so comparing the results of the recurrent network to them was fair and unbiased.

Although we were successful in investigating all our research questions and carrying out our experiments, we still acknowledge that more and varied data would have led to better performing models as well as increased our confidence in the generalizability. We thus believe that if more efforts are made to produce datasets with reformulations, it will create an avenue for better models to be trained and fine-tuned.

Finally, we would like to propose an assumption for future work that could potentially improve the performance of CQR models and even co-reference systems trained on dialogs. This assumption is the ”perfect rewrites” of previous turns. This ”perfect rewrite” assumption states that the previous turns have been perfectly reformulated and thus the slots that should be carried over or entities to be resolved are mostly a turn or two away. This might sound like an unrealistic assumption, but if real dialog systems are observed, we can see that this idea is not too far from the real world. Dialogs between systems are often on a turn basis, and if each turn by the user is reformulated correctly, the dialog system would not need to go to a very early turn to reformulate or resolve queries. This assumption would have an impact on the way the input the sequence to sequence models that would be used to train, develop, and test. Currently, we use a sequence of turns as input, and a reformulation of the last turn by the user as an output. With the ”perfect rewrite” assumption, the new input would be the sequence of reformulated turns, with the last turn of the user not being reformulated while the output would be the reformulated turns. This should not only make the work easier by making models primarily focus on previously reformulated turns rather than going back up to 4 turns away to resolve amphora but could reduce the amount of data needed to be stored as only most recent turns would need to be stored. Storing only a few turns would also have an impact on the speed of the models with little impact on precision and recall.

Appendix A

First appendix

A.1 Architectures

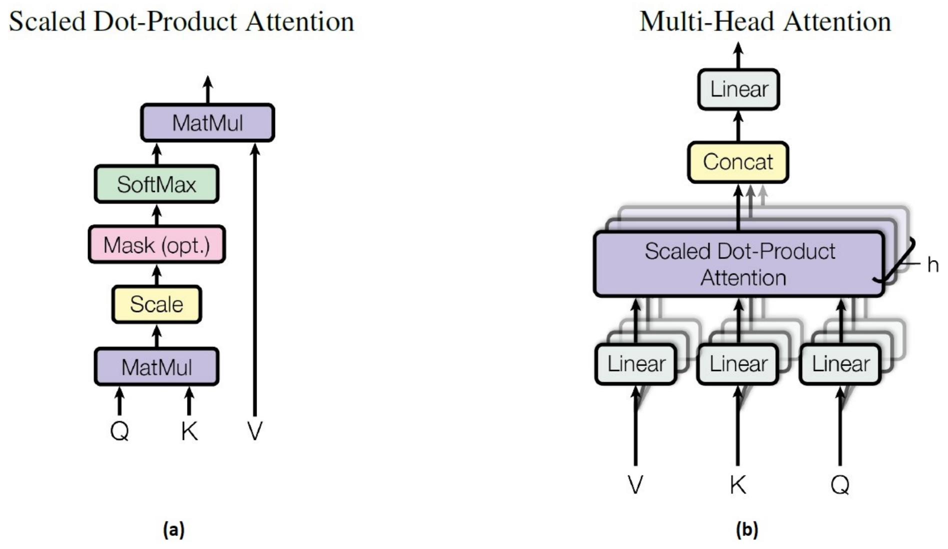


Figure A.1: Scaled Dot-Product Attetion and Multi-Head Attention architectures

Appendix B

Second appendix

B.1 Useful Links

Alexa Dataset
`https://github.com/alexa/alexa-dataset-contextual-query-rewrite`
OpenNMT GitHub
`https://github.com/OpenNMT/OpenNMT-py`
OpenNMT Documentation
`http://opennmt.net/OpenNMT-py/main.html`
Treccast Data
`https://github.com/daltonj/treccastweb/tree/master/2019/data`

B.2 My Code and prepossessed data

All Preprocessed data
`https://github.com/kolaSamuel/Contextual-Query-Reformulation/tree/master/Data`
Results and Analysis
`https://github.com/kolaSamuel/Contextual-Query-Reformulation/tree/master/Data`
Sequence to Sequence Colab
`https://colab.research.google.com/drive/1eEa-ZceY4frTPc66TLrWj71F_16KJu8f`

Bibliography

[1] Mihail Eric and Christopher D. Manning. Key-value retrieval networks for task-oriented dialogue. *CoRR*, abs/1705.05414, 2017.

[2] J. Gao, M. Galley, and L. Li. *Neural Approaches to Conversational AI: Question Answering, Task-Oriented Dialogues and Social Chatbots*. Foundations and Trends in Information Retrieval Series. Now Publishers, 2019.

[3] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[4] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[5] Surafel Melaku Lakew, Mauro Cettolo, and Marcello Federico. A comparison of transformer and recurrent neural networks on multilingual neural machine translation. *CoRR*, abs/1806.06957, 2018.

[6] Surafel Melaku Lakew, Mauro Cettolo, and Marcello Federico. A comparison of transformer and recurrent neural networks on multilingual neural machine translation. *CoRR*, abs/1806.06957, 2018.

[7] Chetan Naik, Arpit Gupta, Hancheng Ge, Lambert Mathias, and Ruhi Sarikaya. Contextual slot carryover for disparate schemas. *CoRR*, abs/1806.01773, 2018.

[8] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[9] Pushpendre Rastogi, Arpit Gupta, Tongfei Chen, and Lambert Mathias. Scaling multi-domain dialogue state tracking via query reformulation. *Proceedings of NAACL-HLT 2019*, pages 97—-10, 2019.

[10] Michael Regan, Pushpendre Rastogi, Arpit Gupta, and Lambert Mathias. A dataset for resolving referring expressions in spoken dialogue via contextual query rewrites (CQR). *CoRR*, abs/1903.11783, 2019.

[11] Hannes Schulz, Jeremie Zumer, Layla El Asri, and Shikhar Sharma. A frame tracking model for memory-enhanced dialogue systems. *CoRR*, abs/1706.01690, 2017.

[12] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368, 2017.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.