UNIVERSITY OF CALIFORNIA
SANTA CRUZ

# GENERATING NATURAL LANGUAGE WITH SEMANTIC AND SYNTACTIC GENERALIZATION

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

**Lena Reed**

September 2021

# Contents

# List of Figures

# List of Tables

## Abstract

Generating Natural Language with Semantic and Syntactic Generalization

by

Lena Reed

Traditional statistical natural language generation (NLG) systems require substantial hand-engineering: many of the components, such as content planners, sentence planners and surface realizers must all be designed and created manually and updated when a new type of utterance is required. Neural natural language generation (NNLG) models, on the other hand, learn to generate text through processing massive amounts of data in end-to-end encoder-decoder frameworks, where syntactic properties are learned automatically by the model. While the learning components of NNLG models are mostly accomplished automatically, they do require, however, that the training data be collected and labeled, which can often be a laborious process. The advantages of not needing handcrafted templates and syntax-to-semantics dictionaries may thereby be offset by the need to retrain neural models with new data as new domains are added, effectively replacing a “knowledge bottleneck” with a “data bottleneck”.

To overcome the data bottleneck, we experiment with methods to leverage existing datasets to allow our NNLG models to generalize to novel meaning representations and sentence planning operations. We explore the generation of artificial data and mixing data from different sources as a way to augment existing training data available to the NNLG. Given our different methods for augmentation, we evaluate whether

NNLGs can learn syntactic and semantic generalizations.

In this thesis, we developed methods to enable NNLG models to perform three types of generalization. First, we generalized multiple stylistic features to a single supervision token which represented the personality formed by the stylistic features. We then generated output with two different personality supervision tokens to create novel personalities not present in the training data. Second, we performed generalization on sentence planning operations. Sentence planning is the module of NLG models that affects the way individual propositions are combined into sentences, which usually has an effect on the final style of the realization. We generalized specific sentence planning operations, such as sentence scoping and contrastive discourse structuring, with values and attribute combinations beyond what was seen in the original training data. Finally, we combined training data from different sources to produce outputs for meaning representations that blend the ontologies from both sources. In all three experiments, we investigated different representations and architectures to enable models to generalize.

Our contributions include the development of methods that enable NNLG models to generalize and thereby expand beyond their original training data. This is an important extension of results to date and a significant step in making NNLG models more useful at low data cost. We also generated and released multiple datasets which we used to train and test our models.

To my family.

## Acknowledgments

I first want to thank my advisor, Lyn Walker, for supporting me through my journey as a PhD student. I'm immensely grateful for all her advice and help. Her love and enthusiasm for the field has been an inspiration and she has helped me grow as a researcher and as a person.

I also want to thank my advancement committee, Lyn, Steve Whittaker, Pranav Anand, Jim Whitehead and Snigdha Chaturvedi for giving me great advice that helped guide me through dissertations and research.

I am also grateful for all the help and support that I received from my labmates and fellow grad students. They have been here for advice and assistance and have helped me through many projects. I want to especially thank Shereen Oraby, who I worked with extensively throughout my degree and who has always been there for me when I needed advice. I also want to thank Wen, Kevin, Jurik, Davan, Brian, Jiaqi, Stephanie, Elahe, Amita, Chao, Geetanjali, and Suzanne.

This dissertation would not have been possible without help from friends and family who assisted me by editing chapters and giving advice. Dhyana, Greg, Mom, and Dad, I appreciate that you took the time to read through my chapters and give me very useful feedback.

Finally, I want to thank my friends and family. You have always been there for me, giving me support and listening to me whenever I needed it. I would not be here without all of you, and I will never be able to express how grateful I am for everything.

I especially want to thank my Mom, my Dad, my sister, Emi, and my step-mom Dani, who have always been there for me.

# Chapter 1

# Introduction

## 1.1 Overview

Natural Language Generation (NLG) is essential for many tasks in Natural Language Processing (NLP), from dialogue systems to question answering to summarization. There are many areas where generating coherent, interesting, and accurate language is necessary to achieve one's goal. While NLG has been fundamental to NLP since its origin [Machinery, 1950], Neural Natural Language Generation (NNLG) has led to many advancements in the field. NNLG uses end-to-end deep neural models to automatically learn from massive amounts of data to create a generator, instead of requiring templates and hand-crafted modules, such as sentence planners and surface realizers, needed by traditional statistical NLG models. End-to-end models, which only require data to learn, allows us to quickly train complex models that can generate a wide range of diverse outputs. Since neural models learn from data with minimal human input, we cannot explicitly control what information is represented by these models. There is still much to learn about how to encourage models handle different tasks and whether specific goals are feasible when employing an NNLG model.

NNLG models depend on exceedingly large datasets, in the range of tens of thousands of sentences, to generate new text. To create new data using an NNLG, we have first to acquire a dataset, traditionally by either finding an existing dataset[1] or develop a dataset oneself either by crowdsourcing or scraping data from online sources and processing it to fit the needed format. Both of these options are very time-consuming and, especially for crowdsourcing, very expensive. But what if one wants to generate new data with only slight variations from an existing dataset? Often generated text needs to be tailored to a specific audience [Paris, 2015; Walker et al., 2004], but being forced to create an entirely new dataset to make minor modifications to the tone of the generated output is impractical.

In this thesis, we develop methods to generalize from existing datasets to create novel output and to control the form and content of the output in order to generate precisely what we need for these tasks. It is commonly assumed that models will generalize beyond their training data. We develop methods that **ensure** that models generalize.

We show that with proper supervision and completely novel methods, such as performing self-training on the model, we can generate utterly novel outputs. We use a combination of existing and synthetic datasets, which makes the data overhead to

[1]while there are some existing and freely available datasets, depending on what you need to generate this might not be possible

generate the desired result relatively small. We systematically experiment with different types of supervision to determine how control features guide the model to generalize to unseen input specifications; we find that even single tokens of supervision can significantly improve the results.

While there has been previous work on generalizing experiments to more complex instances not seen in the training data [Ficler and Goldberg, 2017], these are some of the first experiments in generalization for semantic and syntactic variation in data-to-text NNLG. The experiments focused on explicitly determining how to enable models to generalize to meaning representations (MRs) with combinations of attributes completely novel from the MRs in the training data. We also explore the best ways to encourage these models to learn a general representation of the training data, so the model can learn to generate accurate output given these specific MRs. We hypothesize that NNLG models can generalize with proper supervision, and that baseline MRs would not accomplish these tasks. Throughout this thesis, we will show that for syntactic and style generalization, this is the case. The models can learn to generalize to new values and generate completely novel output with simple token supervision. Still, this is not sufficient for semantic generalization, and we needed to employ a self-training regime that learns from the model's mistakes to achieve our generalization goals.

## 1.2 Generalization of Data-to-Text NNLG

The purpose of these experiments is to establish baselines and then develop methods to allow NNLG models to learn general forms and representations of the data to perform generalization. We can then generate output on MRs that are significantly different from the input training data. The framework of these experiments is data-to-text NNLG in the domain of restaurant reviews. We focus on this type of generation because there are existing and plentiful datasets in the restaurant review domain that can be utilized to develop methods which allows models to generalize across multiple datasets. Also, the data input of the data-to-text generation of these restaurant review corpora allows us to easily add additional control variables since the task already generates text from a series of control variables. Here we will define the task of data-to-text generation in NNLG and provide examples of the datasets. We will go into more detail on the datasets used in these experiments in Chapter 3.

### 1.2.1 Data-to-Text Generation

Data-to-text generation involves taking non-linguistic data and transforming it into text, representing the meaning specified by the input [Reiter and Dale, 2000]. This data often defines the semantics of the desired output but it can also describe other features, such as the length or sentiment of the output. The structure can also have various forms, such as a flat series of attribute-value pairs or a structured tree representation indicating the sentence structure of the output. This type of generation is helpful for any task where one has information that needs to be provided in an easier-to-understand and interpretable manner, such as taking the statistics of a basketball game and creating an output that explains what happened in the game.

The input data, which we refer to as meaning representations or MRs, consists mainly of the semantics of the output. The MR describes the content that we want the output to realize. We also have a limited representation of the discourse structures desired in the output, as well as control features that indicate specific characteristics that we want in the output depending on the experiment, such as a token representing the number of sentences in the generated output. An example of the data is "name[Zorros] cuisine[Italian] decor[good]". The output would need to describe Zorros as a restaurant with Italian food and good decor. This MR has three attribute-value pairs. Every attribute can potentially have multiple values associated with it. Here, *cuisine* is Italian, but it can also be French or Japanese. Content selection and downsampling can be accomplished by not realizing every attribute-value pair in the output. This is a feature in some models and allows them to select the best or most relevant content. In these experiments, we consider an output to be "correct" if it accurately realizes all of the attribute-value pairs in the input. Not including an attribute is therefore considered a mistake. Other mistakes are realizing the incorrect value of the attribute or adding an attribute to the output that was not in the MR. Also, a common mistake that NNLGs make is repeating information, and therefore we consider it a potential semantic error.

Figure 1.1 has an example from the E2E dataset [Novikova et al., 2017b], which we cover in more detail in Section 3.3. In the example, we can see that each value in

| | |
|---|---|
| **name** \|\| Cocum **eat type** \|\| coffee shop **food** \|\| Chinese **price range** \|\| more than £30 **customer rating** \|\| high **family friendly** \|\| no | The Cocum is a coffee shop that serves Chinese food. It is not kid friendly, the rating is high, and the price range is more than £30. |

Figure 1.1: Data-to-Text Example. Each attribute is represented in the MR on the left and the text on the right in the same color.

the attribute is represented in the text. Each attribute has a similar phrasing as the attribute-value pair in the MR, but it does not need to be precisely the same. For example, the natural language phrase “kid friendly” represents the attribute *family-friendly*, which is a similar, but not exactly the same wording. As long as the same meaning is captured, it is correct to describe the attribute and value using this string. The order is also not dependent on the order in the MR. The model reads in the attributes in a fixed order. Each attribute is in the same location in the MR, but it can be represented in the text in any order. Attributes not in the text were not included in the MR. This dataset has one more attribute, location, but since there is no location mentioned in the review, the MR does not have a location either. There are many ways to represent a single MR, and most training datasets have more than one text example per MR to represent this; this is just one example.

### 1.2.2 Restaurant Review Domain

There are multiple datasets in the restaurant domain, examples of which are in Table 1.1. E2E is a crowdsourced dataset where participants wrote restaurant reviews and descriptions based on pictograph representations of the MR [Novikova et al., 2017b]. NYC is a synthetic dataset of restaurant reviews generated with the PERSONAGE [Mairesse and Walker, 2010, 2011] statistical language generator based on actual reviews of restaurants in New York City; this dataset was generated as a contribution of this thesis [Reed et al., 2018]. YelpNLG is a natural dataset scraped from Yelp and processed to automatically produce MRs representing the type of food mentioned, service, and other features [Oraby et al., 2019]. We use the E2E dataset and the NYC dataset in the course of these experiments. We also generate and provide synthetically generated data modeled after E2E. We go into more detail about the specifics of these datasets in Chapter 3.

We use data from the restaurant domain for multiple reasons. The first is that it uses the data-to-text structure that we desire. With this data-to-text structure we can control and evaluate the model's output automatically. Each dataset consists of matched pairs of MRs and text, where the MRs represent the semantics in the text and some limited discourse relations, and the text is in the form of a restaurant review. The structure of the dataset entails that there is often only a finite number of ways to represent the semantics in the output. We can therefore create scripts that automatically calculate the output's semantic accuracy, allowing us to iterate through

| Dataset | MR | Realization |
|---|---|---|
| E2E | name[The Olive Grove]<br>eatType[pub]<br>food[Indian]<br>priceRange[moderate]<br>area[riverside]<br>familyFriendly[no] | There is a pub that is moderate priced called The Olive Grove in the Riverside area, that serves Indian food and is not child friendly. |
| NYC | name[nameVariable]<br>recommend[no]<br>cuisine[Caribbean]<br>decor[bad]<br>food_qual[bad]<br>location[the West Village]<br>price[affordable]<br>service[bad] | nameVariable is the worst restaurant since it provides bad staff and mediocre decor. It offers mediocre food. nameVariable is in the West Village. it's a caribbean place. It is affordable. |
| YelpNLG | food[type:chicken_tacos,adj:pulled,mention:1]<br>food[type:fish_tacos,adj:no_adj,mention:1]<br>food[type:chicken,adj:fried,mention:1 | we also had the pulled chicken tacos, the fish tacos, and the bao with fried chicken. |

Table 1.1: Examples from restaurant review datasets E2E [Novikova et al., 2017b], NYC [Reed et al., 2018], and YelpNLG [Oraby et al., 2019].

multiple experiments quickly. We go into more detail about automatically evaluating the output in Chapter 4.

We also decided to use datasets from the restaurant domain because there are multiple datasets that have similar, but not identical semantics. These experiments across these similar, but different, datasets are the first experiments in this type of transfer learning. To give the model the best chance of success, we explore whether the model can learn to apply sentence operations and attributes to a dataset where they were not seen before, but the two datasets have semantic overlap. For these experiments, having multiple data-to-text datasets in the same domain was crucial.

We also wanted a dataset that had significant stylistic variation. Because the

NYC dataset is synthetically generated and already provides stylistic controls motivated by the Big Five Personality Traits, through NYC, we are able to control many of the linguistic features in the outputs. We are able to generate outputs which represent multiple personality traits, such as disagreeable, agreeable, conscientious, unconscientious and extrovert. Still, we also wanted a crowdsourced natural dataset with interesting syntactic and discourse lexical features, which E2E provides. In the E2E dataset, 5.4% of the training examples use constrastive discourse structures, such as "Near the Rainbow Vegetarian Café is The Rice Boat. It has a low rating but costs less than £20." and 14.5% uses a discourse feature we call fronting, such as "With a 1 out of 5 rating, Cotto is kid friendly, located near Ranch, and serves French food." [Juraska and Walker, 2018].

These datasets also provide variety in the types of values represented in the MR. There are four main types of values across the datasets. The first is a **boolean** value, which is either yes or no. This attribute is represented in the text as a trait that the restaurant has or is explicitly missing. For example, restaurants can either be expressly family-friendly or explicitly not. If a defining feature of the restaurant is neither that it is family-friendly nor not family-friendly, the MR does not include the attribute *family friendly*. The second type of attribute is a **scalar**. Attributes such as *food quality* are scalar, where there are different values, indicating how positively or negatively the author views the restaurant. These values are useful for contrast since one way to contrast two attributes is for them to be explicitly comparable to each other. One cannot contrast two attributes that the author considers positive.

Because these values represent a rating instead of a specific string, there can be multiple ways to realize the value in the output. For example, the utterance can also represent “food_quality[terrific]” with phrases such as “terrific food” or “delicious food”. There are also **finite set** and **infinite set** values. These are very similar. The main difference is that the MR can define finite set values with multiple potential values, such as the attribute-value pair “area[riverside]” can be represented by either of these realizations: “in the riverside area” or “is located near the river”. On the other hand, infinite set values must be represented by that exact string. The attribute-value pair “location[the West Village]” must represent as “the West Village” in the output text because that is the name of the location. Generally, proper nouns must be represented with the exact string that is the value; otherwise, there might be multiple ways to realize the same content.

Finally, these datasets also have grounded real-world applications. For example, rather than just giving categorical ratings, someone who would like to write a review, but does not have confidence in their writing or language skills, could fill in a form with categorical and scalar information and have a review generated automatically. They could then select from multiple possible automatically generated reviews. Also, reviews could be generated in multiple languages. Reviews are often written in the dominant language of the area where the restaurant is located. A system based on a generation engine that could write reviews in another language could help those who do not speak the dominant language learn about the restaurant.

### 1.2.3 Generalization

A common goal of most work in machine learning is the development of models that generalize to unseen test instances. While neural methods in general have led to huge performance improvements for many NLP tasks, much recent work has also been concerned with creating models and methods for better generalization to unseen cases [Chen et al., 2020b; Zhao and Eskenazi, 2018; Goodwin et al., 2020; Chen et al., 2020a]. In the case of data-to-text NNLG, one problem has been that varied and large datasets are needed to ensure the models' semantic accuracy, and review semantic accuracy is often achieved at the expense of stylistic variation. Moreover, our initial experiments show that these models do not generalize well to unseen cases in the way that would be expected.

As mentioned previously, NNLGs depend on large amounts of training data that can generate accurate outputs. By generalizing from existing datasets to unseen cases we will be able generate more diverse output without needing to collect more data. To that end, we developed methods which improve how NNLG's generalize from existing datasets to generate output which has different representations of stylistic, syntactic and semantic features, which are not seen in the training data, with minimal additional supervision.

We developed methods to generalize across three types of generalization, stylistic generalization, syntactic generalization and semantic generalization. Each type of generalization focuses on a different component of the utterances being generated by

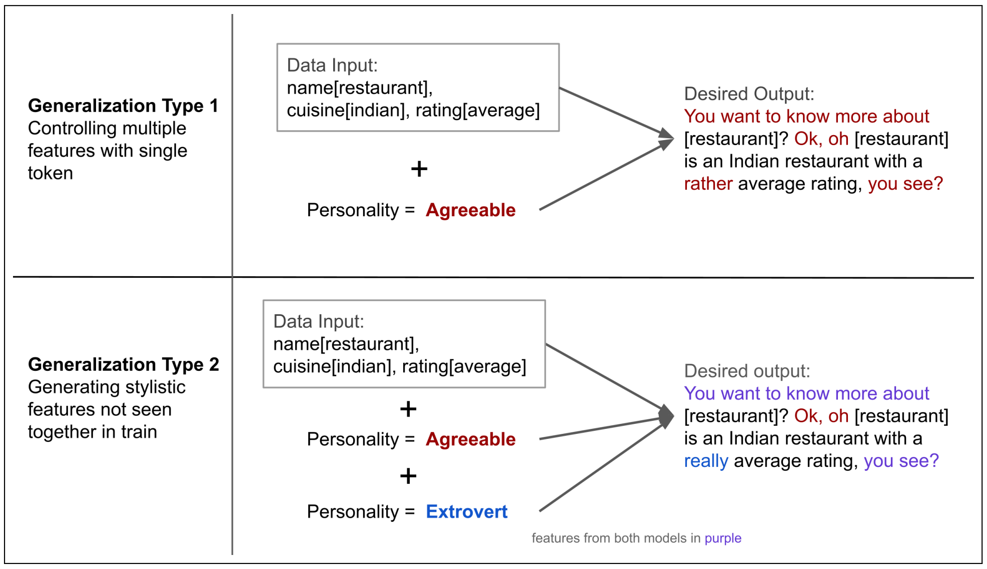
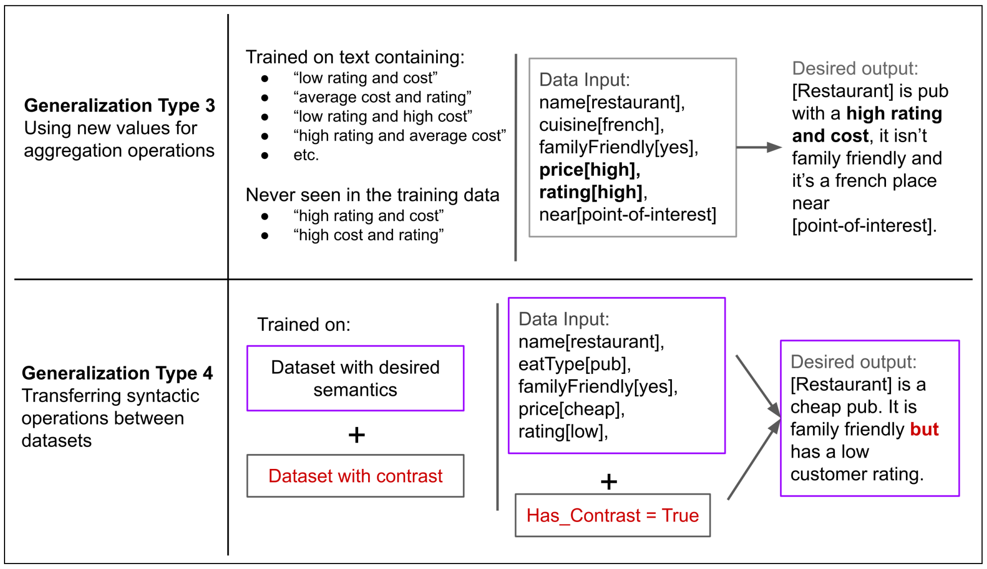

Generalization Type 1
Controlling multiple features with single token
Data Input: name[restaurant], cuisine[indian], rating[average]
+
Personality = Agreeable
Desired Output: You want to know more about [restaurant]? Ok, oh [restaurant] is an Indian restaurant with a rather average rating, you see?
Generalization Type 2
Generating stylistic features not seen together in train
Data Input: name[restaurant], cuisine[indian], rating[average]
+
Personality = Agreeable
+
Personality = Extrovert
Desired output: You want to know more about [restaurant]? Ok, oh [restaurant] is an Indian restaurant with a really average rating, you see?
features from both models in purple

Figure 1.2: Examples of the **stylistic** generalization accomplished in this thesis.

the neural models. Stylistic generalization generalizes stylistic features, features which do not affect the meaning of the utterance, but can affect how the reader perceives the text. Syntactic generalization generalizes across sentence planning operations that combine the attributes in the data input. And semantic generalization generalizes across the semantics being expressed in the generated output. We will now describe in more detail what and how we generalized within these categories.

Stylistic generalization entails generalizing stylistic features present in the data. Stylistic features are important for creating interesting and varied outputs that sound natural. But, neural models often converge on generating the same words and phrases repeatedly. We experimented with two different types of stylistic generalization, which can be seen in Figure 1.2. The first, Generalization Type 1, is controlling multiple features with a single supervision token. Each personality is a combination of many different stylistic features, such as pragmatic markers and aggregation operations. We were able to control the personality generated in the output using a single token which represented that personality.

We also further generalized the stylistic features by generating output with combinations of stylistic features not seen in the training data, Generalization Type 2 in Figure 1.2. Our model, described in the previous paragraph, was trained with five distinct personalities. The token model, which prompted the model to generate a personality given a single token of supervision appended to the input data, was then given **two** personality tokens as supervision when generating output. We determined that this method allows the model to generate completely novel personalities which combined features from multiple personalities.



Generalization Type 3
Using new values for aggregation operations
Trained on text containing:
"low rating and cost"
"average cost and rating"
"low rating and high cost"
"high rating and average cost"
etc.
Never seen in the training data
"high rating and cost"
"high cost and rating"
Data Input: name[restaurant], cuisine[french], familyFriendly[yes], price[high], rating[high], near[point-of-interest]
Desired output: [Restaurant] is pub with a high rating and cost, it isn't family friendly and it's a french place near [point-of-interest].
Generalization Type 4
Transferring syntactic operations between datasets
Trained on:
Dataset with desired semantics
+
Dataset with contrast
Data Input: name[restaurant], eatType[pub], familyFriendly[yes], price[cheap], rating[low],
+
Has_Contrast = True
Desired output: [Restaurant] is a cheap pub. It is family friendly but has a low customer rating.

Figure 1.3: Examples of the **syntactic** generalization accomplished in this thesis.

We then explored syntactic generalization. We had two distinct types of tasks with which we experimented. These syntactic generalization tasks are illustrated in Figure 1.3. The first task, Generalization Type 3, was generating sentence planning operations using values that did not perform those sentence planning operations in the training data. We had two experiments, sentence scoping and distributive aggregation. For sentence scoping, we generated outputs which controlled the number of sentences in the output. We were able to generate the desired number of sentences regardless of the attributes that we were generating. For distributive aggregation, we trained a model where two attributes, *price* and *rating*, were distributed using this an aggregation operation. These two attributes could have one of three values associated with them, high, average and low. In the training data, we only used the values average and low when distributing. The value high was also present in the training data, but did not use the distributive aggregation operation. We were, with a boolean supervision token representing the presence or absence of this operation, able to generate the distributive aggregation operation with the value high. The development of this method allows us to expand aggregation operations to values not seen in train.

We also generalized across datasets by transferring an aggregation operation that was plentiful in one dataset to a dataset which had very few examples, as seen in Generation Type 4 in Figure 1.3. We had one dataset with only a few instances of contrast and we augmented this dataset with another, synthetic, dataset which had contrast in every utterance. This method of augmenting the data and transferring the contrast operation across the two datasets allowed us to generate contrast frequently with high accuracy using a supervision token to encourage the generation of contrast.

Transferring syntactic operations across datasets is an effective method of learning a syntactic operation, along with supervision.

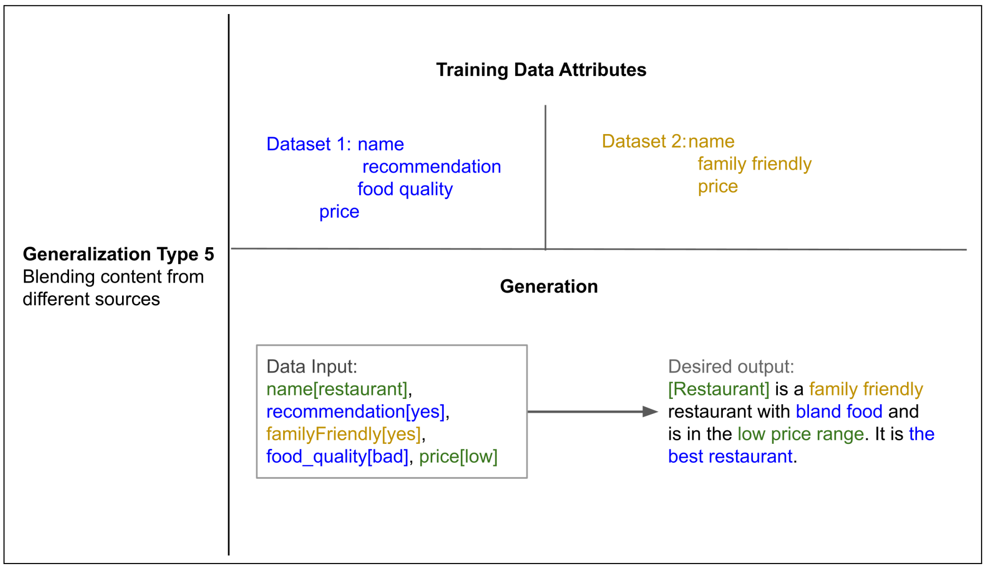


Figure 1.4: Examples of the **semantic** generalization accomplished in this thesis.

The final type of generalization, semantic generalization, involved generating output which blended the semantics of two different datasets. We provide an illustration of this task in Figure 1.4. The two datasets, NYC and E2E, were from the same domain, the restaurant review domain. Though the datasets were from the same domain, they did not have the same attributes. The datasets had four attributes in common and four to five attributes which were unique. We created a new ontology which combined the two two separate ontologies and generated output with blended semantics. We found this was a surprisingly adversarial task, with low semantic accuracy in initial results regardless of supervision method. We developed a self-training regime which improved the results. We achieved a high semantic accuracy on this task, generating output which generalized across the two training datasets.

## 1.3 Summary of Contributions

In this work, we discovered how to control multiple types of generalization using NNLGs. The first is generalizing numerous stylistic features, which create a single personality, into a single "personality" token and generating combinations of stylistic features not seen together in the training data to develop new personalities (Chapter 5). We find that, while the model can control multiple stylistic features with a general "personality" token, more supervision improves the results. Though, we believe that this does not mean that fine-grained control is the best solution as it requires that each feature be explicitly included or excluded when generating. Also, for some personalities, the token supervision does out perform the fine-grained supervision. Controlled generation with a single "personality" token does allow us to generate outputs that blend personalities. This is our second form of stylistic generalization. We controlled the output with two tokens instead of one, generating a new personality not seen in the training data. These two types of generalization are illustrated in Figure 1.2.

Next, we developed supervision and transfer learning schemas which allow for syntactic generalization with different sentence planning operations (Chapter 6). First, we generalized to aggregate new values using these sentence planning operations which were not seen being aggregated with these operations in the training data. Second, we improved the generation of contrast in the output by augmenting a dataset with the desired semantics with data from a dataset with more examples of contrast. We illustrated both of these types of generalization in Figure 1.3. In these experiments we introduced methods to control various sentence planning operation, such as sentence planning and distributive aggregation, when generating using NNLG models. We can control the number of sentences being generated in the output, regardless of the number of attributes in the data input, given a token which either represents the number of sentences or the sentence complexity of the desired output. We also discovered that we can teach a model a general representation of aggregation operations which can be applied to new values that were not seen performing this aggregation operation previously, with a single boolean supervision token. We also proved that datasets with a small amount of a sentence planning operation, in this case, contrast, can learn from other datasets with more examples of the operation and can achieve improved control of the contrast operation by generalizing from one dataset to another.

The final generalization task is combining data from two sources and generating output that combines attributes from each source not seen together in the original datasets (Chapter 7). This generalization task is illustrated in Figure 1.4. This a completely novel task which has not been attempted previously. While results from the initial experiments to blend datasets resulted in low accuracy regardless of supervision method, the model did generate outputs with attributes from both sources The outputs, though they blended the semantics, did not match the input MR. We created an improved model with a learning method that allows the model to learn from its mistakes. We retrofit corrected MRs for the erroneous outputs generated by the model, creating correct, blended, utterances which the model could learn from. This is only possible due to methods for automatic semantic evaluation created for these experiments. This self-training regime lead to significantly improved results. These results demonstrated that we can generate novel outputs from a new ontology. We can, therefore, also potentially use two separate datasets to generate a desired output without needing to develop a completely new dataset to achieve the same results. We also attempted this same experiment with a pre-trained model, GPT-2 [Radford and Narasimhan, 2018], and found that the semantic results did not significantly improve. There were, however, were slight improvements of the amount stylistic variation in the final results after self-training when compared to the results after self-training of the encoder-decoder model used initially.

We also created and released three datasets that were used to perform these experiments.

1. PersonageNLG[2]: 88K MR/utterance pairs of training data and 1,390 pairs of testing data. The utterances are in the restaurant domain and generated using PERSONAGE. They are based on the E2E challenge MRs and utterances. Each utterance has stylistic features which represent one of the following five personalities: agreeable, disagreeable, conscientious, unconscientious and extrovert.

2. Sentence Planning Corpus for NLG[3]: a dataset of approximately 205K MR/utterance pairs demonstrating discourse relations such as contrast and justification, as well

[2]https://nlds.soe.ucsc.edu/stylistic-variation-nlg
[3]https://nlds.soe.ucsc.edu/sentence-planning-NLG

as the use of aggregation operators in NLG.

3. Source Blending in NLG[4]:a dataset of 77K MR/utterance pairs of training data in the restaurant domain from two source ontologies, NYC and E2E and 3040 MRs with attributes from both sources of testing data.

## 1.4 Thesis Outline

This thesis starts by defining the task and components that we will be using in this work. In Chapter 2, we define the models which we used in this work as well as previous work in data-to-text generation and generalization in NNLG. Then, in Chapter 3, we describe the datasets that we used in these experiments; the NYC dataset, generated by the statistical natural language generator PERSONAGE [Mairesse and Walker, 2011], the E2E dataset, a crowdsourced dataset created for the E2E generation challenge [Novikova et al., 2016], and a synthetic E2E dataset, which is also generated using PERSONAGE with the attributes and values from the E2E dataset. In Chapter 4, we go over the evaluation methods that we used to evaluate our results, both standard evaluation metrics, such as BLEU [Papineni et al., 2002] and human evaluation, and the evaluation methods that we designed to evaluate the semantics and stylistic features in the generated outputs. We explain the semantic evaluation metric that we developed, slot error rate or SER, which automatically detect semantic errors in the generated output based on the semantics of the input MR and generates a score

[4]https://nlds.soe.ucsc.edu/source-blending-NLG

based on the number of errors divided by the number of slots in the input. We use this method of automatically detecting errors to also generate MRs from output text, which was vital in creating a self-training regime for our ontology blending task.

The second half of this thesis presents stylistic and syntactic generalization experiments to develop methods to improve NNLG models' ability to generalize across various tasks and types of generalization. In Chapter 5, we evaluate if models can learn to generate multiple types of stylistic features with a single token of supervision or if more fine-grained control is needed to accomplish the task. In Chapter 6, we test how models can generalize to new values when learning aggregation operations and if aggregation operations can generalize across multiple datasets. In Chapter 7, we experiment with techniques to teach models how to make semantic generalization across two datasets. This allows the models to generate output that combines attributes never seen together in the training data. Finally, the conclusion chapter, Chapter 8, summarizes the contributions of this thesis and describes future directions.

# Chapter 2

# Previous Research

## 2.1 Overview

The goal of this thesis is to develop methods that enable Neural Natural Language Generation (NNLG) models to better generalize to unseen test instances. In this chapter, we will start by going over NLG methods and architectures and NNLG models. Then, we will explore recent work done in different generalization tasks, such domain transfer.

## 2.2 Natural Language Generation

NLG is the subfield of NLP which involves data-to-text generation and text-to-text generation. Data-to-text generation generates from either a semantic or database representation and text-to-text generates from text input; they both generate text as output. This is a wide area of research, with many types of generation, such as machine translation, question answering, text summarization and dialogue generation [Garbacea and Mei, 2020]. Inputs for NLG tasks can be anything from text to structured knowledge to an image. In this research, we focused on data-to-text NLG. In the datasets used in this thesis, the input is a flat representation of the semantics that are desired in the output. We are experimenting with *how* NNLG models generate text, and ways to control this generation, given an input meaning representation or MR. First, we will review NLG more broadly, including the components of NLG and the architectures, and then we review over data-to-text generation in particular.

### 2.2.1 NLG Tasks

There are six basic tasks found in many template based, grammar-based and statistical NLG systems. These tasks generally describe what steps need to be accomplished by any NLG system, either explicitly or implicitly [Reiter and Dale, 1997, 2000].

1. Content Determination: the process of deciding what information should be in the text.

2. Discourse Planning: the process of deciding the order in which the information should be presented in the text.

3. Sentence aggregation: the process of grouping data into separate sentences and how data will be combined in a single sentence, if grouped into a single sentence.

4. Lexicalization: the process of deciding specifically which words and phrases will be used in the text.

5. Referring expression generation: the process of deciding which words and phrases will be used to express domain entities.

6. Linguistic realisation: the process of creating the final output with correct grammar.

These six stages of generation are sometimes accomplished in a separate module for each task. Though, in many models, these stages are combined, either all into a single step or into multiple steps that each accomplish one or more stages. For example, a system could both decide what content will be realized and the order in which it will be realized in a single module of the generator. These six stages are often completed explicitly in statistical NLG models. NNLG models, on the other hand, often complete all these steps implicitly. They do not have explicit modules to perform each of these stages. However, even with models that accomplish these tasks implicitly, all of these must be executed to take in input and generate text. Controlling the decisions being made in each of these stages is needed to generalize operations to new, unseen, inputs.

### 2.2.2 NLG Architectures

There are three dominant approaches to NLG architectures, which can also be described as ways that NLG systems can be organize the tasks listed above [Gatt and Krahmer, 2018]. These are as follows:

1. Modular architectures: the tasks are divided amongst different modules where each module does one or more particular task.

2. Planning perspectives: these are similar to the modular approach but with less strict boundaries between subtasks.

3. Integrated approaches: these are not divided into tasks and are usually dependent on statistical learning.

Modular architectures are the oldest methods of NLG. They are often considered pipe-lined architectures, with each task handled by a module in the pipeline. The module completes the task which it was designed for and then the output of that module is passed to the next module in the pipeline. Realizing the final, textual output is the final step. Different models can divide the tasks up differently between modules. Some models have multiple tasks in a single module and some even split up a single task between multiple modules.

Planning based approaches, on the other hand, often do not divide these different tasks cleanly across different modules. In a planning perspective, goals are accomplished by identifying a sequence of actions that are needed to accomplish the goal. This means, that if needed, the "what to say" and "how to say it" tasks, which are clearly different tasks in modular architectures, can be combined and approached simultaneously. Planning, for example, can be done through grammar by viewing linguistic structures as planning operators. Since these structures both indicate what is being said and the rules which indicate where it can be said, tasks such as sentence planning and realisation are not separated into different modules.

Finally, integrated, or end-to-end, approaches do not break the generator into different modules, tasks or actions. There are many ways to approach this kind of generation, including using context-free grammar rules learned from data using statistical machine translation methods [Wong and Mooney, 2007]. These end-to-end methods also include deep learning neural methods, which we go into more detail in Section 2.3.

### 2.2.3 Data-to-Text Generation

Data-to-text generation is a classic NLG problem [Kukich, 1983; Reiter and Dale, 1997] that takes structured data and generates natural language. The structured input data can come in many forms, such as a records or tables [Mei et al., 2015; Wiseman et al., 2017] or a set of triples [Gardent et al., 2017], but the output must be natural language. The goal of the generator is therefore to take the input data and generate a fluent output where the information in the text transcribes some or all of the information in the input data. This can have many applications, such as sport reports [Chen and Mooney, 2008], presenting customized scientific information [Wanner et al., 2015], and interacting with financial data [Plachouras et al., 2016].

The boundaries of what can be done in data-to-text generation with state-of-the-art NNLG models is constantly being expanded. While there has been speculation that these models allow for end-to-end training that does both the sentence planning and surface realization steps mentioned previously [Dusek and Jurcícek, 2016b; Nayak et al., 2017], recent work has found that even when using neural methods a more pipe-lined approach leads to more fluent output [Ferreira et al., 2019].

Several different approaches have examined more hybrid architectures, such as in work by [Jiang et al., 2020]. In this work, they experimented with combining both explicit representations of sentence planning operations with neural methods to focus on the most relevant data in the data structures presented to the model. There has also been work which defines multiple modules for content and sentence planning while maintaining the end-to-end nature of neural models [Puduppully et al., 2019]. Semantic errors have also been explored in recent work, such as Shen et al. [2020], where they take advantage of the segmental nature of data-to-text generation and create a model which explicitly must generate each section of the input data as a segment in the output text. There is also work that both modifies the architecture and the input data. In Rebuffel et al. [2021], they aligned each word in the output text to the tokens in the input and the decoder of the model leverages these alignments to generate fewer hallucinations.

There has been recent work in data-to-text generation to create new datasets, such as DART, a large dataset of RDF triples in multiple domains [Nan et al., 2020] and ToTTo, a table-to-text dataset that produces a single sentence description from a highlighted set of cells from a Wikipedia table [Parikh et al., 2020]. Existing datasets such as E2E [Dušek et al., 2020], RotoWire [Wiseman et al., 2017], and WebNLG [Gardent et al., 2017] have been cited and used hundreds of times since their releases [Fu et al., 2019; Zeng et al., 2018; Fu et al., 2019; Balakrishnan et al., 2019; Kedzie and McKeown, 2019; Puduppully et al., 2019]. Data-to-text generation is a rapidly growing field and researchers need to be able to generate new output for different tasks and situations, without necessarily needing to create an entirely new dataset.

## 2.3 Neural Natural Language Generation

NNLG models are behind recent most advances in NLG. They are able to automatically learn from raw data to represent features and model complex high-dimensional distributions. Essentially NLG systems can be created with minimal supervision and hand-crafted work, when one has a significant amount of training data. The steps covered in Section 2.2.1 are all combined into a single end-to-end framework. It is commonly assumed that every step is automatically learned by the model, without the need for intermediate representations, allowing significantly less hand-engineering and the ability to learn new tasks by creating new datasets [Wen et al., 2015; Dusek and Jurcícek, 2016b; Nayak et al., 2017; Wen et al., 2016].

### 2.3.1 Seq2seq models

Sequence-to-sequence, or seq2seq, models [Sutskever et al., 2014] are popular in many NNLG tasks. This is because they do not require a known and fixed length input, and the length of the input and output can be different. Seq2seq models have two components, an encoder and a decoder. They map a sequence input onto a fixed dimensionality vector in the encoder, which then gets passed to the decoder where the target sequence is decoded using that vector. This is very useful for data-to-text generation where very frequently the data and text are not the same length. Seq2seq models are conditional language models, also referred to as encoder-decoder models because they consist of two main parts, an encoder which interprets the input, and a decoder, which decodes the output. The conditional probability of the output given the input can be expressed as:

$$p(y_1, y_2, ..., y_{T'}|x_1, x_2, ..., x_T) = \prod_{t=1}^{T'} p(y_t|v, y_1, y_2, ..., y_{t-1}) \tag{2.1}$$

where $x$ is the input sequence, $y$ is the output sequence, $T$ is the length of the input and $T'$ is the length of the output.

The encoder and decoder of seq2seq models are often composed of either Recurrent Neural Networks (RNNs) [Rumelhart et al., 1986; Mikolov et al., 2010] or Long Short Term Memory units (LSTMs) [Hochreiter and Schmidhuber, 1997]. RNNs are a neural network architecture that is able to model long-term dependencies by passing the output of the feedforward network which processes each item of a sequence as input to the next item in the sequence. But while RNNs should be able to model any length of sequence, the reality is that vanishing and exploding gradients means that they cannot actually consider context beyond a few previous items [Bengio et al., 1994].

LSTMs are the solution to this problem and therefore are widely used in seq2seq models. LSTMs have a similar architecture to an RNN, except with an additional memory cell with a self-connected recurrent edge which stores information over many timesteps. At each timestep the LSTM cell takes as input the item in the sequence at that timestep and the hidden output and the internal encoder state of the LSTM cell from the previous timestep and output a hidden output and an internal encoder state [Dušek, 2017]. The memory cell $c_t$ has a node with an internal hidden state $h_t$ and a series of gates. The gates regulate what information is retained and what is forgotten in the cell using a sigmoid neural net layer. The first gate is the input gate $i_t$, which controls what values and how much the unit is updated. The second gate is the forget gate $f_t$, which controls how much information from the previous cell is forgotten. The third and final gate is the output gate $o_t$, which controls how much of this cell is output. The operations computed by the LSTM at each timestep are [Graves, 2013]:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)\\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)\\
c_t &= f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)\\
o_t &= \sigma(W + xox_t + W_{ho}ht - 1 + W_{co}c_t + b_o)\\
h_t &= o_t tanh(c_t)
\end{aligned}
\tag{2.2}
$$

where $\sigma$ is the logistic sigmoid function, $x_t$ is the input at timestep $t$ and $W$ are learned weight matrices. Using these formulas, an LSTM can control what information is remembered and forgotten allowing the gradient to pass through without vanishing or exploding.

Both the encoder and the decoder are composed of either RNNs or LSTMs. The seq2seq model then learns from the training data, which is a set of paired input and outputs, using maximum likelihood to maximize the conditional log-likelihood of the correct output given the input for each instance in the training data. For NNLG, in order to generate an output given only an input, there are different decoding methods that can be selected from in order to generate the most likely output sequence.

To generate the output, the decoder scores a potential sequence using the following equation:

$$\hat{y} = arg\,\underset{y}{max}\, P(y|x) = arg\,\underset{y}{max} \prod_{t=1}^{N} P(y_t|y_{<t}, x) \tag{2.3}$$

where each word in the output sequence, $y_t$ is scored given the previous words that were predicted and the input $x$. The probability distribution over the next word is commonly calculated using a softmax function.

To generate the most likely output, the decoder, theoretically, would calculate the probability of every potential output sequence and then select the one output with the highest likelihood. But calculating the probability of every possible combination of words is not feasible, since the problem is exponential in length. This means that we use different decoding strategies to prune off branches that are unlikely to yield the best result. While there are multiple strategies that accomplish this, for all of the experiments and models, we use beam search. Beam search is a breadth-first search method over the hypothesis space, where for each token the model only keeps the $k$-best hypotheses where $k$ is the beam size. This method has proven to be effective and has become the standard algorithm for many NLG tasks.

All of the models that we used in this thesis also use an attention mechanism [Bahdanau et al., 2014; Luong et al., 2015]. This mechanism is used to enhance seq2seq models by allowing them to learn to pay attention to particular sections of the sequence, especially with longer utterances where not all of the input necessarily informs the token that is currently being generated. Attention for each target token is calculated over all the tokens in the input and higher values are assigned to tokens that are considered relevant for the token being generated.

In this thesis we used multiple seq2seq frameworks based on the state-of-the-art and the availability of NNLG models. For the experiments in Chapters 5 and 6, we built the models using the open-source framework TGen [Dusek and Jurcícek, 2016a], which is implemented in Tensorflow [Abadi et al., 2016]. For the experiments in Chapter 7, we used the open-source seq2seq modeling framework OpenNMT-py [Klein et al., 2017], which is implemented in pyTorch [Paszke et al., 2019]. We also used GPT-2, a pre-trained model. We used an open-source implementation of GPT-2 available on Huggingface. Pre-trained models are the current state-of-the-art in NNLG, we will go into more detail about GPT-2 and other pre-trained models here.

### 2.3.2 Pre-trained Models

Since the introduction of BERT [Devlin et al., 2019] in 2018, there has been a growing interest in using pre-trained models in NNLG. Pre-trained language models are trained on large amounts of unannotated data, which allows the model to learn the general structure of human language, including features like word usage and grammar. There have been many pre-trained models released in the last couple of years, including GPT-2 [Radford and Narasimhan, 2018], GPT-3 [Brown et al., 2020] and ELMo [Peters et al., 2018]. After pre-training, the model can be fine tuned for particular tasks with smaller amounts of data, since it only needs to learn the specific task, instead of also needing to learn how to generate human language in general. Since the data fed to the model does not need to be annotated, millions or billions of utterances can be used to train the pre-trained model. This has allowed BERT, and other pre-trained models, to excel at tasks such as question answering [He et al., 2020], sentiment analysis [Li et al., 2020; Karimi et al., 2020] and named entity recognition [Liang et al., 2020]. Experiments using and improving these models are being released in new papers and new models are appearing very frequently.

Models are pre-trained using an assortment of tasks, which can be summarized into three categories: supervised learning, unsupervised learning and self-supervised learning [Qiu et al., 2020]. Supervised learning involves learning to map an input to an output and requires input/output pairs. Unsupervised learning tasks learn from just unlabeled, unpaired data and find intrinsic knowledge from the data. Finally, self-supervised learning is a blend of the two, where the learning paradigm is the same as supervised learning, but no labels or data pairs are needed since the model creates those automatically.

There are multiple types of tasks within these three categories that can be used to pre-train a model. The most common unsupervised task is probabilistic language modeling, where the model learns the probability of the next word given all of the previous words. The product of the probabilities for each word is the overall probability of that utterance. This can be accomplished either in one direction or two directions. When carried out in two directions the model is a bidirectional language model (LM), which is a combination of two unidirectional LMs, one for left-to-right and one for right-to-left.

Another common pre-training task is masked language modeling, which was based on the cloze procedure [Taylor, 1953]. This task is a self-supervised task where the model masks some tokens in the utterance by replacing the actual token with a special mask token or a random token. The model then learns to predict these tokens based on the rest of the model. This task was adapted for BERT [Devlin et al., 2019]. There have also been improvements made to the basic masking procedure, such as dynamic masking where a different mask is generated each time the data is fed through the model [Liu et al., 2019]. Also, there are additional tasks used for pre-training such as permuted language modeling [Yang et al., 2019b] and denoising autoencoder [Lewis et al., 2020].

Pre-training is hypothesized to be a type of regularization and can help the model avoid overfitting [Erhan et al., 2010]. This means that pre-training has been shown to help models avoid generalization errors. This is why we experimented with this type of model, to see if the general knowledge of language gained from pre-training will aid in these attempts to generalize on specific datasets.

GPT-2 [Radford and Narasimhan, 2018] is one of the most recent pre-trained models that is widely available and used. We experiment with generalization using GPT-2 in this thesis. GPT-2 is a transformer-based language model, which is different from a seq2seq model because it does not have an input and an output. Instead the language model predicts each next word based on the words given so far, $P(y_t|y_1, ..., y_{t-1})$. While these models are generally useful for generating text such as in creative writing, etc., they can be used for other types of generation. For the system that we employed, we are able to generate output text by training the model on data and text concatenated with an end token and then testing by only passing the input data. Since the model predicts the next word based on all previously seen words, it is able to learn to generate the review text given the data input.

## 2.4 Generalization in Neural Natural Language Generation

Generalizing to new input has always been a feature desired in Natural Language Generation (NLG). If we can only generate instances that we have seen in the training data then we are very limited in what we can generate. While simply generalizing to unseen data is standard in NLG tasks, and test sets are generally composed of entirely unseen data, this is just the first step in generalization. To really generalize, we need to be able to generate instances which are significantly different from the data seen in the training data. This is how we will be able to leverage existing datasets to combat the data bottleneck and learn to generate on small amounts of data. This is not an easy task. Generalizing, especially to adversarial examples, has been shown to be a challenging task that neural models often fail to accomplish [Wallace et al., 2019; Feng et al., 2018; Ribeiro et al., 2018; Goodfellow et al., 2014]. Generalizing is a growing field of research, the different techniques to approach these problems appearing often.

The data bottleneck is widely believed to be a fundamental issue within NNLG with work being accomplished to combat it in different ways[Oraby et al., 2019; Chen et al., 2020b; Chang et al., 2021]. The data bottleneck is a result of NNLG needing thousands of utterances to train, and that annotated datasets take a tremendous amount of time and resources to create. When it takes 5hrs to label 50 utterances, creating tens of thousands of utterances to train a model on a new domain is not feasible [Peng et al., 2020].

Pre-training appears to be the main way that recent work has approached the problem of generalization and tackling the data bottleneck. There have been multiple papers using pre-trained models to perform few-shot tasks. Few-shot means that the model only needs to be trained or fine-tuned on a small amount of data to learn new tasks, instead of the usual large amount of data needed to train a neural model. In Chen et al. [2020a], they use a pre-training regime that exploits unlabeled data-to-text data to create a generation model that is designed to generate knowledge-enriched text. They found that they could achieve the same results as a baseline model with only one fifteenth of the data.

Generalization across domains has been the goal of multiple recently released papers, and pre-trained models are the most recent attempt at solving this problem. While we are not experimenting with domain transfer in this work, these few-shot tasks are examples of how existing datasets are being leveraged to generate novel output. New models are being generated that can more easily learn few-shot tasks, such as Chen et al. [2020b] where they created a model to generalize across domains. Their model is designed in two phases, the first performs content selection on the input data, and the second step generates using the pre-trained GPT-2 language model to generate coherent text. This allows them to improve on the baseline with just 200 training examples. In Peng et al. [2020], they also want to more easily learn to expand to new domains, in this case, in the context of task-oriented dialogue generation . This few-shot NLG task involves created a pre-trained model and fine-tuning an small amounts of data within the target domain. They create a model, based on GPT-2, called SC-GPT, which is pre-trained on a large amount of dialogue act labeled utterances as well as the GPT-2 pre-training to help it learn the data-to-text task-oriented dialogue generation problem. With this model they are able to generalize to new domains with a small number of training instances in the target domain.

Data augmentation is another way to handle a lack of training data. In this work, we generated synthetic data based on existing datasets, changing the stylistic features present in the training data using a statistical generator. In Chang et al. [2021], they also generate new data from existing data-to-text datasets using two different methods. The first is by replacing values which are in both the input data and the output text with other values in the same category and the second is by using GPT-2 and cycle consistency to generate new training instances. With these data augmentation techniques with less than 10% of the annotated data they outperform seq2seq models trained on 100% of the annotated data.

## 2.5 Summary

In this chapter, we described NLG, NNLG and generalization within NNLG. In Section 2.2, we went over the tasks needed to generate text, including content planning, sentence planning and realization. We also described the different types of architectures that can do these tasks. This work is focused on data-to-text generation and in this chapter we describes the state-of-the-art in data-to-text generation tasks. In Section 2.3, we went over the architecture of NNLG models, more specifically seq2seq models and pre-trained models. Then, in Section 2.4 described the state-of-the-art in work in generalization, including work using pre-trained models such as GPT-2 [Radford and Narasimhan, 2018] and few-shot learning tasks.

# Chapter 3

# Creating Corpuses

## 3.1 Overview

The goal of this thesis is to test how well NNLG models' generalize. Being able to generalize will allow us to generate different types of data with existing datasets and help users get more use out of a single dataset, which will help combat the data bottleneck in NNLG. To perform these tests, we needed datasets which we used to train the NNLG models. In this chapter, we will describe the three datasets that we use in the experiments and the methods used to create them. We used three data-to-text datasets from the same domain, the restaurant review domain. We decided to use datasets within the same domain so that we can learn to generalize within and across datasets.

The three datasets that we use are all in the same domain, restaurant reviews. They are data-to-text datasets, so each instance in the dataset is a meaning representation (MR) and utterance pair. The MR is made up of a series of attribute and value pairs which represent the semantics in the utterance. The utterance is one or more sentences which represent the attributes and values in the MR in the form of a restaurant review. The three datasets are E2E, E2E synthetic and NYC. E2E is a crowdsourced dataset [Novikova et al., 2017b], and E2E synthetic and NYC are synthetic datasets. E2E synthetic is a synthetic dataset based off of the original E2E dataset, so they have the same attributes and values. We will describe the process of creating a synthetic dataset and the NYC dataset in particular in Section 3.2, the process by which E2E was created and the attributes and values in the dataset in Section 3.3, and converting the E2E dataset into a synthetic dataset Section 3.4.

We hypothesized that synthetically created data, which is automatically generated using Statistical Natural Language Generation (SNLG), can be used to augment crowdsourced data. This is why having both synthetic and crowdsourced datasets from the same domain is vital for these experiments. Synthetically generated data can be generated quickly without having to query humans for utterances, a task which takes time and money. It is also very structured, so if you want significant amounts of data with a specific feature, one can acquire that dataset by generating synthetic data. This is true even if that type of construction is quite rare in datasets generated by people. Synthetic text data is used to both perform specific sentence planning generalization and to combine data from multiple sources. As mentioned previously, crowdsourced data generally takes significant time and capital to collect and the quality of the data is difficult to control. On the other hand, crowdsourced data can also be more stylistically varied than synthetic data, though this can be difficult to elicit from the human annotators. There is also a third type of dataset, natural datasets. These are the most stylistically varied and complex of these datasets, extracted from existing sources on the internet, e.g. Twitter and Yelp, but they do not have semantics associated with them. They therefore need to be generated either automatically or with complex crowdsourcing strategies, which is why we do not use them in these experiments for data-to-text generation. Both synthetic and crowdsourced datasets will be used in this thesis.

There are three types of datasets: (1) synthetic, which is automatically generated using statistical natural language generation (SNGL), (2) crowdsourced, which is generated by asking humans to write sentences given instructions or another type of prompt, and (3) natural, which is gathered from sources, generally online, where people write at will (Yelp, Twitter, blogs, etc). In the data-to-text generation tasks, each dataset has its advantages and disadvantages.

### 3.1.1 Synthetic Datasets

Synthetic datasets can be generated quickly and cheaply with a statistical generator, which we define in Section 2.2.2. Statistical generators, given enough potential content, can create a dataset with as many utterances as needed. Statistical generators can perform content planning and sentence planning allowing the models to decide the order of the content, the aggregation operations used to combine them, additional stylistic features to add to the utterance to modify the voice and other modifications. A set of just a few templates can therefore create hundreds of unique utterances, depending on the number of voices and the potential variety of features that can be added to contribute to the voice. We are then able to generate synthetic datasets of tens of thousands of utterances, which are needed to train an NNLG model.

Synthetic datasets are especially useful for creating controlled datasets with particular stylistic features. When generating a dataset with a statistical generator one has complete control over the types of features present in the dataset. For example, if one wants to test a NNLG's ability to generate contrast correctly, a statistical generator can generate thousands of examples of contrast and the NNLG can be trained and the model's ability to generate this particular feature can be tested. It also makes testing these features more feasible without needing the resources to generate a crowdsourced dataset with the specific intention to include this feature.

There are a few disadvantages of using a completely synthetic dataset. One issue is that this method relies on templates and any new attribute needs a new template. These templates, which will be explained in more detail in Section 3.2.1, need to have the syntactic structure of the phrase so that the SNLG can add in adverbs and other features while maintaining grammaticality. Since these templates are not simple to generate it takes an expert annotator to generate them. Also, each stylistic feature that needs to be controlled needs to be added to the SNLG if not already present, complete with the rules needed to add the feature and maintain the fluency and grammar of the utterance. The model also needs as input the probability of each feature appearing in the text and these probabilities need to be hand-tuned. While this is all often less work than generating a crowdsourced dataset or collecting a natural one, it is not an insignificant amount of effort. This also means that there are limited sentence structures and the text will not have the stylistic variation that exists in natural text. Also, since the statistical model must use hand written rules and templates to generate, grammatical errors are possible. The utterances might also, even if grammatically accurate, sound artificial and lack the fluency that is present in utterances written by a human. Any grammatical errors or unnatural phrasing might be learned by the NNLGs trained on this dataset, so the model can potentially generate these mistakes as well. This is one reason why when evaluating the outputs of the NNLG model with human annotators it is important to also evaluate the synthetic dataset so that a proper baseline can be established.

### 3.1.2 Crowdsourced Datasets

Crowdsourced datasets are more natural than synthetic data since humans write the sentences that are in the dataset, though, as mentioned previously, it is much more expensive and time consuming to generate these types of datasets. To generate a crowdsourced dataset for a data-to-text task, the annotators are presented meaning representation (MR), either in a written or visual format, and then asked to write an utterance based on the MR and a series of instructions. These prompts will inform the author if there are other guidelines they need to follow, such as the type of utterance they need to write (review, statement, etc.), if there are any length restrictions (maximum or minimum number of words or sentences), etc.

An advantage of the crowdsourced dataset is that it allows the authors to control the instructions and the data given to the crowdworkers and the type of data they generate. If the authors want a specific feature in their dataset they simply need to add instructions requesting that the crowdworkers include those features in their writing and they have a dataset with the needed feature. Also, for data-to-text datasets, since the MR was used to generate the dataset the authors do not need retroactively design an MR and match it to data. There will be a fixed set of attributes and values and the utterance will already be matched to a complete MR. If the quality of the crowdworkers' results are properly controlled the utterances should have accurate grammar and should be fluent and natural since humans wrote the utterances.

While these advantages may make it seem like crowdsourced datasets are optimal for data-to-text generation, there are still some downsides. The first is that it is expensive and time consuming to generate these crowdsourced datasets since tens or hundreds of thousands of utterances are necessary to train an NNLG and it can take a minute or two to generate a single utterance. If one assumes it will take one minute to generate an utterance, and the goal is to pay crowdworkers 8.5USD an hour, it would cost approximately 1500USD to get 10,000 utterances. Collecting a new dataset for every new attribute or feature one wants to add becomes prohibitively expensive. Also, in order to get varied sentences, one does not want to influence the writers by including too much text in the semantics presented to them, so a schema must be created to allow for variety. Finally, while the text will most likely be more grammatically accurate and natural, it is hard to get humans to generate truly varied text using crowdsourcing, since most participants are more likely to generate the minimum requirements presented to them to try to maximize the number of tasks they can complete, since in most cases for these crowdsourced tasks the more utterances a single crowdworker can generate the more money they will make.

### 3.1.3 Natural Datasets

Natural datasets are datasets that are gathered from existing sources, generally online, where humans are generating written utterances not for the purpose of the dataset, but instead for multiple different reasons, such as writing reviews and blogs. While there are many natural datasets for different NLP tasks, such as classification [Waseem and Hovy, 2016] and sentiment analysis [Socher et al., 2013], there are fewer for natural language generation. One reason for this is that while there are many different sources of natural language data, the task of NNLG needs pairs of data, either text-to-text pairs or data-to-text pairs, both of which are difficult to find online. For text-to-text pairs, some tasks are easier to find natural datasets for, such as summarization [Lu et al., 2020], though others, such as changing the tone of an utterance, are much more difficult to collect pairs for. Data-to-text datasets are challenging to extract from online sources since there generally is not a fixed set of attributes that can appear in the text. The authors are not bound to certain attribute or values in their writing, and even with more concrete tasks such as writing reviews, there are many different features of the entity they are reviewing that the reviewer can write about. One dataset of data-to-text, YelpNLG, generates meaning representations (MRs) by selecting attributes that they would find in the text, such as food words, and then generating MRs for reviews which had these features present [Oraby et al., 2019]. Additional features in the text were not included in the MR.

These natural datasets have many advantages, they are cheaper to collect than crowdsourced data, they are often very varied and often the writers' tone and voice are apparent in the utterances. They are the most natural of the three types of datasets, synthetic datasets often are clearly written by a statistical generator and crowdsourced datasets are created by crowdworkers who are trying to get as many utterances written as quickly as possible, so neither are perfect representations of how people create utterances when doing so of their own volition. The main downside of these natural datasets is that they are not designed for NNLG. MRs are often inaccurate or only capture part of the utterance. Because in this work we are testing the bounds of generalization and the ability of models to learn very specific attributes in the utterance, while natural datasets have the most variety and represent what we hope NNLGs will eventually be able to generate, there are not currently useable natural datasets for these tasks.

## 3.2 NYC Dataset

The first dataset we will describe is the NYC dataset. This is a synthetic dataset, which consists of data in the form of attributes and values paired with utterances, generated by the Statistical Natural Language Generator (SNLG) PERSONAGE [Mairesse and Walker, 2007]. The utterances are reviews of restaurants and the data which generates the reviews comes from real reviews of restaurants in NYC, which is why we refer to the dataset as NYC. In this section we will describe the attribute/value pairs used to generate the utterances in the dataset, then we will go over the the SNLG PERSONAGE which generates the dataset before finally describing the particulars of the dataset itself.

#### 3.2.0.1 Source of Data

The NYC dataset is from the PERSONAGE's built in content pool, which is based on a database of restaurants in New York City. This dataset consists of approximately 1000 restaurants and the information was gathered from information freely available from the internet [Walker et al., 2002b]. There are seven different attributes, four of which have scalar values associated with them, and, therefore, can be used for contrast and concession. Each restaurant has six attributes associated with it, which describe the qualities of the restaurant. There are four quantitative attributes, *food quality*, *service*, *decor* and *price* and two categorical attributes, *cuisine* and *location*. Each of the four scalar values are transformed to map from 0-100 in their original ratings, where higher values are more desirable. Food quality, service and decor are mapped directly since their original values also had higher values as more desirable. The inverse was taken for price, so the highest listed price was mapped to 0, since higher price values are generally less desirable. Since these values are from real data, the reviews are semantically feasible, for example, poor food quality and excellent service is a combination that appears very infrequently in the data, as there are not many actual restaurants with this combination of attributes. This allows us to use these combinations with the knowledge they make semantic sense. An example output from PERSONAGE using all the attributes in Table 3.3 is in Row 1 of Table 3.2.

One challenge we face in generating diverse output in NNLG is that NNLG models are highly sensitive to the distribution of phenomena in the training data, and the personality experiments showed that the outputs of NNLG models exhibit less stylistic variation than their training data [Oraby et al., 2018b]. Moreover, even large corpora, such as the 50k E2E Generation Challenge corpus, may not contain particular stylistic variations. For example, out of the 50k crowdsourced examples in the E2E corpus, there are only 1,956 examples of contrast with the operator *but*. There is also only one instance of distributive aggregation because attribute values are rarely lexicalized identically in E2E. In NYC on the other hand, attributes food quality, service, price and decor all have scalar values that allow for contrast, and we are able to increase the probability of any stylistic variation we want to occur more frequently so we can create a dataset with ample contrast. The rhetorical structure also means that we have a dialogue act, *recommend*, that supports PERSONAGE ability to have a *justify* relation, which is in the NYC example in Row 1 of Table 3.2 with the text "I know xname is alright because". The rest of the reference is informing this *recommend* act.

### 3.2.1 Personage Tool

PERSONAGE is a highly parameterizable statistical generator which uses deep syntactic structures (dsynts) to generate outputs with different stylistic variations such as pragmatic markers and aggregation operations [Mairesse and Walker, 2011]. A dsynt is what represents each attribute in the dataset, and describes how to generate a textual output for that attribute. PERSONAGE takes an input two files, a textplan, which indicates which dsynts can appear in the reference and their relationship with each other, such as *justify* and *infer*, and a parameter file which contains the probability of each feature, such as types of aggregation operations or pragmatic markers, being used by the model. PERSONAGE can create many different outputs for each textplan given, depending on the variety of aggregation operations and pragmatic markers in the parameter file as well as the number of dsynts in the textplan.

The architecture for PERSONAGE is in Figure 3.1. The model has three modules which it uses to take the input files and output natural language. These three modules are the content planning module, the sentence planning module and the realization module. The content planning module selects and structures the content, making decisions such as content size, polarity and content ordering. This is where the generator does both content determination and discourse planning, where the dsynts are selected and ordered. Next is the sentence planning module, which has four submodules. The first is syntactic structure selection, where it decides features like the number of first person pronouns that will be realized when generating the dsynt based on inputs like syntactic complexity and polarity. Next, the second and third submodules perform aggregation and pragmatic marker insertion respectively and will be covered in more detail below. The fourth and final submodule within sentence planning, lexical choice, selects which words will be used in the output. The realization module combines all the decisions of the previous two and outputs the natural language utterance.

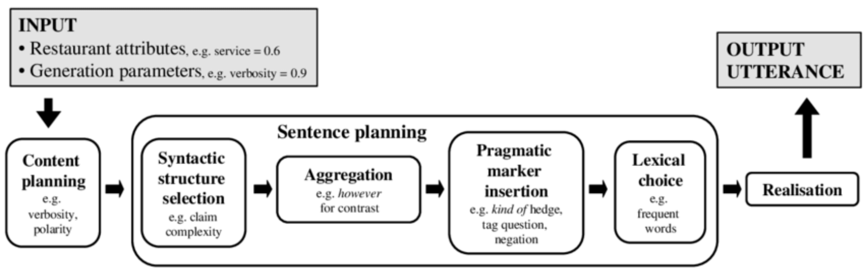
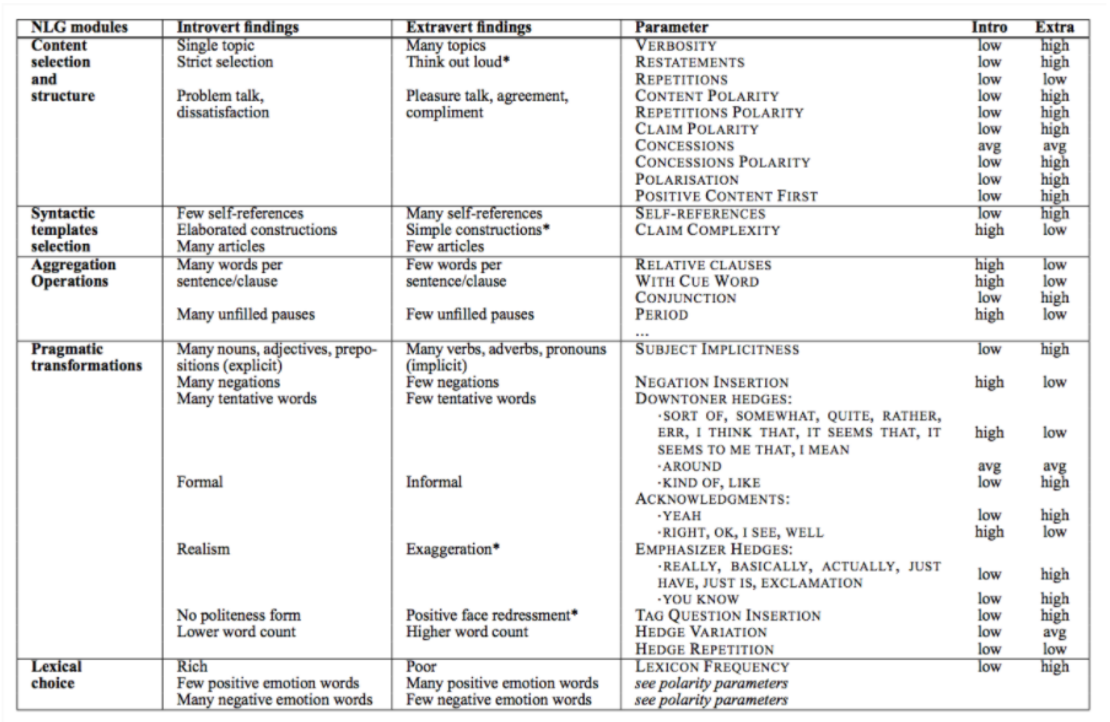

INPUT
• Restaurant attributes, e.g. service = 0.6
• Generation parameters, e.g. verbosity = 0.9
Content planning e.g. verbosity, polarity
Sentence planning
Syntactic structure selection e.g. claim complexity
Aggregation e.g. however for contrast
Pragmatic marker insertion e.g. kind of hedge, tag question, negation
Lexical choice e.g. frequent words
Realisation
OUTPUT UTTERANCE

Figure 3.1: The architecture of the PERSONAGE generator.

The pragmatic markers and aggregation operations defined in the parameter file and used by PERSONAGE can be tuned manually to form the desired output, but there are also built-in personality templates for the ten personalities from the Big Five personality traits. We have used these personality templates to generate data with significant stylistic variation. An example of the stylistic variation used for the introvert and extrovert personalities and the rationale behind including each one in either the extrovert or introvert personalities is in Figure 3.2. While there are many features for each personality in both the content planning phase of the model and in all four steps of sentence planning, not all are used in a single utterance. Given a parameter file with all these features PERSONAGE decides which features to use in the utterance using the structure in the dsynts. For example, an output utterance from PERSONAGE is "xname is a Japanese place near xnear, you know, it is a pub mate and it is in the city centre!". The dsynts are realized in a single sentence and there are two emphasizer hedges, "you know" and an exclamation. But other features, like tag questions and the downtoners "kind of" and "like" are not present because of the length of the utterance.

A list of the primary aggregation operations and pragmatic markers used by



| NLG modules | Introvert findings | Extravert findings | Parameter | Intro | Extra |
|---|---|---|---|---|---|
| **Content selection and structure** | Single topic | Many topics | VERBOSITY | low | high |
| | Strict selection | Think out loud* | RESTATEMENTS | low | high |
| | | | REPETITIONS | low | low |
| | Problem talk, dissatisfaction | Pleasure talk, agreement, compliment | CONTENT POLARITY | low | high |
| | | | REPETITIONS POLARITY | low | high |
| | | | CLAIM POLARITY | low | high |
| | | | CONCESSIONS | avg | avg |
| | | | CONCESSIONS POLARITY | low | high |
| | | | POLARISATION | low | high |
| | | | POSITIVE CONTENT FIRST | low | high |
| **Syntactic templates selection** | Few self-references | Many self-references | SELF-REFERENCES | low | high |
| | Elaborated constructions | Simple constructions* | CLAIM COMPLEXITY | high | low |
| | Many articles | Few articles | | | |
| **Aggregation Operations** | Many words per sentence/clause | Few words per sentence/clause | RELATIVE CLAUSES | high | low |
| | | | WITH CUE WORD | high | low |
| | | | CONJUNCTION | low | high |
| | Many unfilled pauses | Few unfilled pauses | PERIOD | high | low |
| | | | ... | | |
| **Pragmatic transformations** | Many nouns, adjectives, prepositions (explicit) | Many verbs, adverbs, pronouns (implicit) | SUBJECT IMPLICITNESS | low | high |
| | Many negations | Few negations | NEGATION INSERTION | high | low |
| | Many tentative words | Few tentative words | DOWNTONER HEDGES: | | |
| | | | ·SORT OF, SOMEWHAT, QUITE, RATHER, ERR, I THINK THAT, IT SEEMS THAT, IT SEEMS TO ME THAT, I MEAN | high | low |
| | | | ·AROUND | avg | avg |
| | Formal | Informal | ·KIND OF, LIKE | low | high |
| | | | ACKNOWLEDGMENTS: | | |
| | | | ·YEAH | low | high |
| | | | ·RIGHT, OK, I SEE, WELL | high | low |
| | Realism | Exaggeration* | EMPHASIZER HEDGES: | | |
| | | | ·REALLY, BASICALLY, ACTUALLY, JUST HAVE, JUST IS, EXCLAMATION | low | high |
| | | | ·YOU KNOW | low | high |
| | No politeness form | Positive face redressment* | TAG QUESTION INSERTION | low | high |
| | Lower word count | Higher word count | HEDGE VARIATION | low | avg |
| | | | HEDGE REPETITION | low | low |
| **Lexical choice** | Rich | Poor | LEXICON FREQUENCY | low | high |
| | Few positive emotion words | Many positive emotion words | *see polarity parameters* | | |
| | Many negative emotion words | Few negative emotion words | *see polarity parameters* | | |

Figure 3.2: The personality model for the INTROVERT and EXTROVERT personalities.

PERSONAGE can be seen in Table 3.1. PERSONAGE uses the dsynt's structure and values to determine when an aggregation operation can be used to concatenate two dsynts using an appropriate operation [Stent et al., 2004]. The "with" relation, for example, has two syntactic contrasts: (1) the subjects of the clauses must be identical and (2) the clause being concatenated using the "with" must have a have-possession predicate. While this operation has very specific requirements, others, like merge, only need one identical argument and period has no requirements.

Examples of output from PERSONAGE can be seen in Table 3.2. In Rows 1&2 there are examples of the narrator voice, from the NYC source and the E2E source (to be covered in more detail in Section 3.3) respectively. These utterances realize the

| **Attribute** | **Example** |
|---|---|
| AGGREGATION OPERATIONS | |
| PERIOD | *X serves Y. It is in Z.* |
| "WITH" CUE | *X is in Y, with Z.* |
| CONJUNCTION | *X is Y and it is Z. & X is Y, it is Z.* |
| ALL MERGE | *X is Y, W and Z & X is Y in Z* |
| "ALSO" CUE | *X has Y, also it has Z.* |
| PRAGMATIC MARKERS | |
| ACKNOWLEDGE_DEFINITIVE | *right, ok* |
| ACKNOWLEDGE_JUSTIFICATION | *I see, well* |
| ACKNOWLEDGE_YEAH | *yeah* |
| CONFIRMATION REQUEST | *let's see what we can find on X, let's see ....., did you say X?* |
| INITIAL REJECTION | *mmm, I'm not sure, I don't know.* |
| COMPETENCE MITIGATION | *come on, obviously, everybody knows that* |
| FILLED PAUSE STATIVE | *err, I mean, mmhm* |
| DOWN_KIND_OF | *kind of* |
| DOWN_LIKE | *like* |
| DOWN_AROUND | *around* |
| EXCLAIM | *!* |
| INDICATE SURPRISE | *oh* |
| GENERAL SOFTENER | *sort of, somewhat, quite, rather* |
| DOWN_SUBORD | *I think that, I guess* |
| EMPHASIZER | *really, basically, actually, just* |
| EMPH_YOU_KNOW | *you know* |
| EXPLETIVES | *oh god, damn* |
| NEAR EXPLETIVES | *oh gosh, darn* |
| IN GROUP MARKER | *pal, mate, buddy, friend* |
| TAG QUESTION | *alright?, you see? ok?* |

Table 3.1: Examples of aggregation operations and pragmatic markers.

content of the MRs but have very few other features, though there is some variety in aggregation operations, with a contrast in Row 1. Rows 3-7 have examples of six of the different personalities that PERSONAGE can generate. The pragmatic markers that are added to the references are in bold. Some personalities, like EXTROVERT, do not have many pragmatic markers that are added to the text, and others, such as AGREEABLE, have multiple. There is also variety in aggregation operations with personalities for example unconscientious has multiple uses of "also" while disagree has none.

PERSONAGE also allows for variety in its output by having multiple templates per dsynt, one of which is randomly chosen using probabilities that are input into the dsynt. This allows for some templates to occur more frequently than others. Also, dsynts can have synonyms included for words in its structure, which PERSONAGE can select from. Decor, ambiance and atmosphere are all synonyms that PERSONAGE can chose from in the decor dsynt. When inserting a value into the dsynt one can also give a list of options and allow the generator to choose from them. So if one wants to say that the food is great the system can choose between a list of words such as amazing, fabulous, tasty, etc.

With this tool we are able to create synthetic data which we can use to either train an NNLG with crowdsourced data that has been augmented by synthetic data, or to test whether or not specific features are generalizable using NNLG. By creating the data synthetically we do not need to crowdsource the data, which takes more time and resources, but instead we can create thousands of output references with a variety of different stylistic features that we control. We have made synthetic sets of data for different personalities that will be described in Section 5.3.1, for distribution of adjectives across nouns in Section 6.4 and contrast data in Section 6.5, as well as 38k examples using the narrator voice that will be described in Section 7.2.

| | Domain | Personality | MR | Utterance |
|---|---|---|---|---|
| 1 | NYC | NARRATOR | name[xname], recommend[yes], cuisine[Italian], decor[good], qual[good], location[TriBeCa SoHo], price[expensive], service[acceptable] | I know xname is alright because it is pricey with friendly service, but it provides flavorful food though. It is in TriBeCa SoHo. xname is an Italian restaurant. It offers good decor. |
| 2 | E2E | NARRATOR | name[xname], eatType[coffee shop], food[Indian], priceRange[more than £30], customer rating[high], familyFriendly[yes] | xname is a coffee shop with a high rating, xname costs more than £30 and it's kid friendly and an Indian place. |
| 3 | E2E | AGREE | name[xname], eatType[pub], priceRange[high], customerRating[average], area[riverside] | **Did you say xname? Well it is** a pub with a **quite** average rating and it is expensive kind of in riverside, **alright?** |
| 4 | E2E | DISAGREE | name[xname], eatType[coffee shop], food[French], priceRange[cheap], customerRating[excellent], familyFriendly[yes] | **Obviously, basically** it's a French place and cheap with a **damn** excellent rating and xname is a coffee shop. It is kid friendly. |
| 5 | E2E | CONSC | name[xname], eatType[pub], customerRating[average], familyFriendly[no], near[xnear] | **Let's see what we can find on xname. I see, I think that** it is a pub with an average rating and it isn't family friendly, also it's **sort of** near xnear. |
| 6 | E2E | UNCONSC | name[xname], eatType[coffee shop], food[Italian], priceRange[£20-25], customerRating[high], familyFriendly[no] | **Oh God I don't know. I mean xname** isn't kid friendly, also the rating is **darn high**, also it is a coffee shop, also it is an Italian place. It has a price range of £20-25. |
| 7 | E2E | EXTROVERT | name[xname], food[Italian], priceRange[moderate], area[city centre], familyFriendly[no] | xname is an Italian place in city centre and moderately priced, also it isn't family friendly, friend, **you know!** |

Table 3.2: PERSONAGE Natural language outputs with different personalities.

### 3.2.2 Generated Dataset

The NYC dataset was generated using the meaning representations described in beginning of Section 3.2 and the model PERSONAGE described in Section 3.2.1. We required a basic dataset without specific personalities (as mentioned above, PERSONAGE can generate pragmatic markers and aggregation operations based on personality). To do this, we generated the data using very basic pragmatic markers and set the probability of the basic aggregation operations to be equal to each other. This means that while there are some pragmatic markers, such as "like", they do not appear too frequently in the dataset. Each textplan we used to generate an utterance had all six attributes that describe the restaurants in the NYC dataset. PERSONAGE then generates a subset of these attributes in the output utterance. There are exactly 779 different combination of attributes describing 779 different restaurants. We had 1558 different textplans, two for each restaurant with a positive and negative recommendation associated with each of them. PERSONAGE ran 50 times per textplan and we collected the output for every successful run. Since PERSONAGE is a statistical generator, in some cases the model fails to generate an output. Each textplan generated 15-50 utterances. We then needed to generate the meaning representations (MRs) for each utterance.

For each utterance, though we knew the potential attributes and values that could be in the utterance, we needed to detect the attributes which were generated because PERSONAGE does not always generate all the attributes. Since each attribute can only be generated a finite number of ways, given the dsynt used to generate it, we can automatically detect which attributes are in the utterance and generate the MR. We also, for some of the attributes, modify the values in the textplan when generating the MR. While for *food quality*, *service*, *decor* and *price* the values given in the textplan are numerical, when realized by PERSONAGE different numbers fall into different buckets which determines which words are generated to describe the attribute. To make it easier for the models to learn to generate we transformed the scalar values into categories. Since these textplans are reviews, PERSONAGE always generates the name of the restaurant and the recommendation dialogue act and therefore is automatically included in the MR. We then simplified the generation task by delexicalizing the name of the restaurant in both the MR and the utterance. We delexicalized the name (for example "Cottos") by replacing it in the textplan with the variable "xname". The attributes and potential values for each attribute can be seen in Table 3.3.

| **Attributes** | **Values** | **Scalar** |
|---|---|---|
| name | xname | No |
| food quality | excellent, good, decent, bad, terrible | Yes |
| service | excellent, good, decent, bad, terrible | Yes |
| cuisine | French, Italian, Indian, Chinese, Thai... | No |
| location | Manhattan, Harlem, Upper East Side... | No |
| price | cheap, affordable, expensive, very expensive | Yes |
| decor | excellent, good, decent, bad, terrible | Yes |

Table 3.3: NYC attributes and values.

When we finished generating the utterances and MRs we had 76,823 MR/utterance pairs. Repeated MR/utterance pairs were not removed from the dataset, so there were 70,219 unique references and 7,009 unique MRs. Example MR/utterance pairs from the synthetically generated NYC dataset that we use in the experiments in this thesis can be seen in Table 3.4. Stylistic features generated by PERSONAGE can also be seen in this table, demonstrating that there are different features that appear in the dataset. There are also many aggregation features, with 48,272 utterances with the "with" aggregation operation, and the average number of attributes per sentence is 2.54.

## 3.3 E2E Dataset

The E2E dataset[1] is from the E2E NLG challenge [Novikova et al., 2017b], which was created to test the capabilities of end-to-end data-driven NLG methods. It is a data-to-text dataset where the utterances are reviews or descriptions of restaurants. The full data set consists of about 50K references with their MRs. This is then split into two separate datasets, a training dataset and a development dataset, with 42K reference-MR pairs in the training data and 5K reference-MR pairs in development. An additional 630 MRs form the test set, with an average of 7.5 references per test set MR for evaluation. The MRs in each set do not appear in the others. The test set was quite adversarial since the MRs on average were significantly larger than the MRs in the training and development set, can be seen in Table 3.5.

### 3.3.1 Creation of Dataset

The E2E dataset was created through crowdsourcing using Crowd-Flower. In order to generate high quality data while crowdsourcing they employed some techniques

[1] `http://www.macs.hw.ac.uk/InteractionLab/E2E/`

| **MR** | **Utterance** | **feature** | **N** |
|---|---|---|---|
| name[xname] recommend[no] cuisine[New_American] decor[acceptable] food_qual[good] location[the_West_Village] price[expensive] service[acceptable] | Because xname is expensive and a new american restaurant, **but** it provides tasty food though. It is in the West Village with friendly staff and acceptable decor, I wouldn't suggest it. | contrast | 8716 |
| name[xname] recommend[no] cuisine[Korean] food_qual[acceptable] location[The_Bronx] price[affordable] service[bad] | **Right**, I believe you wouldn't love xname because it is a korean restaurant with rude staff. It is in The Bronx. this place provides decent food. it's affordable. | acknowledge definitive | 890 |
| name[xname] recommend[no] decor[bad] food_qual[acceptable] service[bad] | **I see**, xname is the worst place with acceptable food, bad staff and mediocre decor. | acknowledge justification | 925 |
| name[xname] recommend[no] location[Manhattan] price[affordable] service[acceptable] | **Yeah**, I wouldn't suggest xname since it's affordable and this place is in Manhattan with satisfying service. | yeah | 462 |
| name[xname] recommend[yes] cuisine[Spanish] decor[acceptable] food_qual[acceptable] location[Manhattan] price[cheap] service[bad] | I would suggest xname since it is **kind of** in Manhattan with acceptable food and pleasant ambiance. xname is a Spanish restaurant. even if it is cheap, it provides rude staff. | kind of | 541 |
| name[xname] recommend[no] food_qual[acceptable] | xname is **quite** the worst place with kind of adequate food. | general softener | 1649 |
| name[xname] recommend[yes] cuisine[French] location[Chelsea] price[affordable] | **I guess** xname is alright because it is affordable and a french place. it's in Chelsea. | down subord | 563 |

Table 3.4: NYC utterances with most common narrator voice stylistic features. N=76,823

|  | **Number of Attributes in MR** |  |  |  |  |  |
|---|---|---|---|---|---|---|
| **Dataset** | **3** | **4** | **5** | **6** | **7** | **8** |
| TRAIN | 0.13 | 0.30 | 0.29 | 0.22 | 0.06 | 0.01 |
| TEST | 0.02 | 0.04 | 0.06 | 0.15 | 0.35 | 0.37 |

Table 3.5: Percentage of the MRs in the training and testing data in terms of number of attributes in the MR.

designed to avoid areas where crowdsourcing can fail. First, they provided clear instructions, complete with MR/utterance pairs. The workers received the average pay for the platform to not encourage unqualified workers to cheat. And, finally, they employed a validation procedure where they pre-validate the workers to ensure they were fluent in English and checked that at least 20 seconds was spent generating the utterance and they performed human evaluation on the collected data [Novikova et al., 2016]. The participants were given visual representations of the data so that they were not influenced by the word choice of the MR and so they were more likely to produce more natural and informative utterances. An example of a pictorial MR is in Figure 3.3

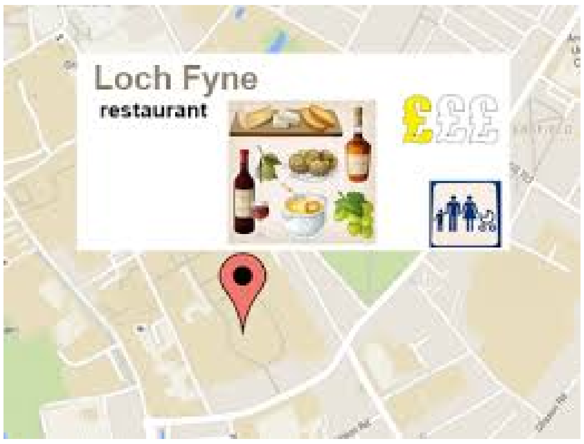


Figure 3.3: Example of a pictorial MR.

The E2E dataset has eight different attributes, each of which have at least two potential values. The attributes *name* and *near* have by far the most options for values, so we generally delexicalize them, so they each have only one value option, “xname” or “xnear” respectively. After that the attribute with the most number of values is food with eight, which represents the type of food the restaurant serves. See Table 3.6 for a list of attributes and their values.

| **Attributes** | **Values** |
|---|---|
| name | xname |
| area | city centre, riverside |
| eatType | coffee shop, pub, restaurant |
| familyFriendly | Yes, No |
| food | Fast Food, Chinese, English, French, Indian... |
| near | xnear |
| priceRange | less than £20, more than £30, cheap, high... |
| rating | 1 out of 5, 5 out of 5, average, high... |

Table 3.6: E2E attributes and values.

The E2E dataset is also more lexically rich than previous datasets, with a mean segmental type-token ratio (MSTTR) value of 0.75. It is also lexically sophisticated, otherwise known as lexical rareness, with 57% of the words in the references not on the list of 2,000 most frequent words generated from the British National Corpus. It has a diverse set of bigrams and trigrams with 61% of trigrams and 50% of bigrams only appearing once in the dataset. The D-Level Scale [Lu, 2014] was also used to evaluate the syntactic variation and complexity of the references. 46% of the data are simple sentences, such as “The Loch Fyne is a moderately priced family restaurant”, but most of the data has a more complex structure, such as “The coffee shop Wildwood has fairly priced food, while being in the same vicinity as the Ranch" and 16% of the data has the highest levels of syntactic complexity with sentences such as "Serving cheap English food, as well as having a coffee shop, the Golden Palace has an average customer rating and is located along the riverside" [Novikova et al., 2017b]. Also, analysis of the data shows that there are interesting discourse markers in the data that make it stylistically diverse. For example, contrast appears in 5.4% of the training data, and gerunds appear in 11.2% of the data[Juraska and Walker, 2018].

## 3.4 Synthetic E2E Dataset

We also created a synthetic version of the E2E dataset where we could control the features present in the utterances. For this dataset, we created dsynts to generate synthetic data based on the E2E dataset. We used these dsynts to create multiple additional synthetic E2E datasets with specific features. These dsynts have one template for generating each of the attributes in Table 3.6, which creates a dataset much more structured than the original E2E. Also, because of this structure, the MRs are more accurate, every attribute in the MR is realized in the reference for that MR since we can correct the MR to match the output of PERSONAGE. We can also add the pragmatic markers and aggregation operations from PERSONAGE to create references that have more stylistic variation, though not the same types as are present in the crowdsourced version of this dataset. Examples of synthetic E2E references are in Table 3.2 in Rows 2-8.

We created three synthetic E2E datasets for the experiments in this thesis, which we will briefly go over here and explain in further detail in later chapters. First we created a personality dataset, with 88k MR/utterance pairs in the training data and 1390 pairs in the testing data. This dataset uses five of the personalities built into PERSONAGE and all the features associated with them. We generated this data set to test NNLG models' ability to learn stylistic features and generalize to labels which represent multiple features. The second dataset is 64k MR/utterance pairs with different numbers of sentences to test NNLG's ability to learn sentence scoping. Here we used the disagreeable personality since it has the highest likelihood of aggregating using a period and then removed all pragmatic markers from the utterance. Finally we have the distributive training dataset, with 64k MR/utterance pairs, where we apply specialized aggregation operations on the attributes price and rating. We go into more detail on this dataset in Chapter 6.

## 3.5 Summary

In this section we went over the different data-to-text datasets that we use throughout these experiments. Since NNLGs depend on massive amounts of data to train they are a vital part of this thesis. All the datasets are in the restaurant domain, allowing us to combine data from multiple datasets without losing semantic coherency and by using synthetic datasets we can cheaply generate data with specific features, allowing us to do concise experiments on very particular attributes in an academic setting.

The three main sources of data that we covered in this section are the NYC dataset, a synthetic dataset generated using the statistical natural language generator PERSONAGE, the E2E dataset, a crowdsourced dataset designed for an end-to-end generation challenge, and a synthetic E2E dataset, which again uses PERSONAGE but uses novel dsynts which were designed to match the attributes seen in the E2E dataset. We go into more detail on the exact datasets used in each experiment in Chapters 5, 6 and 7.

# Chapter 4

# Evaluation Methods

## 4.1 Overview

Neural Natural Language Generation (NNLG) is difficult to evaluate, since language itself is difficult to judge for "accuracy". Natural Language Generator (NLG) outputs are not obviously correct or incorrect like the outputs to classification problems. Instead, there are often multiple ways to generate output which can be considered correct. In both data-to-text and text-to-text NLG, the model can generate text selecting different orderings of content, employing synonyms when selecting words, choosing to insert different pragmatic markers, and making many more decisions which will change the output of the model, while still achieving the desired goal.

Two classic ways to evaluate the output of all types of NLGs are statistical comparison to reference utterances and human evaluation. These both have their advantages and disadvantages, which we will go over in this chapter. Four of the most common metrics which evaluate NLG outputs by comparing the generated utterance to a reference utterance are BLEU, NIST, METEOR, and ROUGE_L. One problem with these metrics is that they are all dependent on having a reference utterance to compare with the generated output. In the following experiments we do not always have reference utterances, making these metrics not only unreliable but also unusable.

Human evaluation also has its flaws. Beyond just being time consuming and expensive, it can often be hard to prompt humans to accurately rate the features of the output being rated. There has been recent attempts to standardize and improve human evaluation in NLG [Belz et al., 2020; Howcroft et al., 2020] but currently human evaluation is not necessarily the gold standard that many of us might desire. Additionally, while we stated that there are flaws beyond the amount of time and money it takes to get human evaluation, those are not insignificant concerns. It is especially concerning when using NNLG models. Researchers need to be able to test and compare many models with slight variations in parameters, supervision and datasets. Needing to stop and get human evaluations after each iteration of training is not a viable option if we want to compare more than one or two models.

To this end, we define the slot error rate (SER), which is modeled after word error rate. Word error rate is commonly used in speech recognition and machine translation systems. Slot error rate has been used previously in work such as [Wen et al., 2015], though they define the errors in their slot error rate, or EER, as the number of missing and redundant slots in the output. We define four distinct types of errors that we believe cover all the possible ways to incorrectly realize the data in a meaning representation (MR). We go into more detail on this, the main metric for evaluating semantic accuracy in the output, in Section 4.4.

The experiments also rely on analyzing various stylistic features, such as those mentioned in Section 3.2.1. These are representations of different features which we are controlling in the MR. We must determine that these features are in the output as expected and use scripts to analyze the text and compare it to the input MR. The scripts then either calculate the accuracy or the correlation on the test set.

## 4.2 Automatic Evaluation

Automatic evaluation has always been a goal within the field of NLG. Human annotators are expensive and can take significant amounts of time. Therefore there have been multiple attempts in past research to create reliable automatic evaluation methods that allow researchers to evaluate the outputs of their models quickly and cheaply.

In the early 2000s, four evaluation metrics for NLG were released that are still used frequently today, BLEU, NIST, METEOR and ROUGE_L. All four compare the outputs of the model to gold standard test references. Because these metrics only require gold standard test references, all four can be used in data-to-text and text-to-text NLG, provided there are gold standard test references for comparison. While these metrics have been useful at providing quick evaluation for NLG models to allow efficient testing of different parameters and inputs, they are not perfect metrics which capture everything one would want in NLG outputs. Here we will first go over the different automatic evaluation metrics we use in this paper. We then will explain why they are not sufficient and why we created evaluation metrics to evaluate the results.

The first evaluation metric, one of the most widely used metrics in NLG, is BLEU [Papineni et al., 2002]. It was designed to evaluate the output of machine translation models, although it is widely used in various types of NLG today. This metrics compares the output to one or more references using a modified $n$-gram precision, where the maximum number of times any one $n$-gram can be counted positively towards the precision in the output is the maximum number of times that $n$-gram appears in any one reference. As a result, utterances with unwanted repetition of $n$-grams that appear in the references are not rewarded for their repetition. The metric then combines the $n$-gram precision using a geometric mean which takes into account that the precision decays exponentially as $n$ increases. The metric also penalizes short utterances so that outputs that only realize a subsection of the output do not get higher scores than outputs that realize more of the reference strings with a few errors. The final equation for BLEU is:

$$BLEU = BP * exp(\sum_{n=1}^{N} w_n log p_n)$$

where $BP$ is the brevity penalty, $w$ is the positive weight and $p$ is the modified $n$-gram precision. BLEU was one of the first attempts at using automatic metrics to evaluate NLG models. Though it does have problems, on which we will go into detail, most papers on NLG require BLEU scores to be complete.

Another automatic metric often used to evaluate NLG is NIST [Doddington, 2002]. NIST is modeled off of BLEU. While in BLEU each $n$-gram is weighted evenly, NIST weights the $n$-grams by importance. It calculates this weight by counting the number of times the $n$-gram occurs in the references over the total number of reference $n$-grams and taking the log of this frequency. The less frequent the $n$-gram, the more important it is.

Another metric that takes BLEU and modifies it is METEOR [Banerjee and Lavie, 2005]. METEOR tries to explicitly address some weaknesses of both BLEU and NIST. These weaknesses are lack of recall and the use of higher order $n$-grams to measure grammaticality instead of measuring this explicitly. Also, the use of geometric averaging of $n$-grams, which can result in a score of zero for a single sentence if one of the component $n$-gram scores is zero. METEOR corrects these flaws by mapping between references and candidates by first looking at exact matches, then stemming the words and mapping, and finally mapping using Wordnet synonyms. It then finds the precision and recall over the entire dataset and calculates a final combined score by calculating the F-score of the precision and recall.

The final automatic evaluation metric from previous work is ROUGE_L [Lin and Och, 2004]. ROUGE_L was designed to improve upon the progress made by metrics such as BLEU. The authors believed three main points: (1) that the brevity penalty used by BLEU was insufficient as a replacement for recall, (2) that the higher order n-grams are not a substitute for considering sentence level structure and (3) that the geometric mean can cause a score of zero incorrectly for some sentences. They modified BLEU by including a recall using the longest common subsequence of the reference and the candidates.

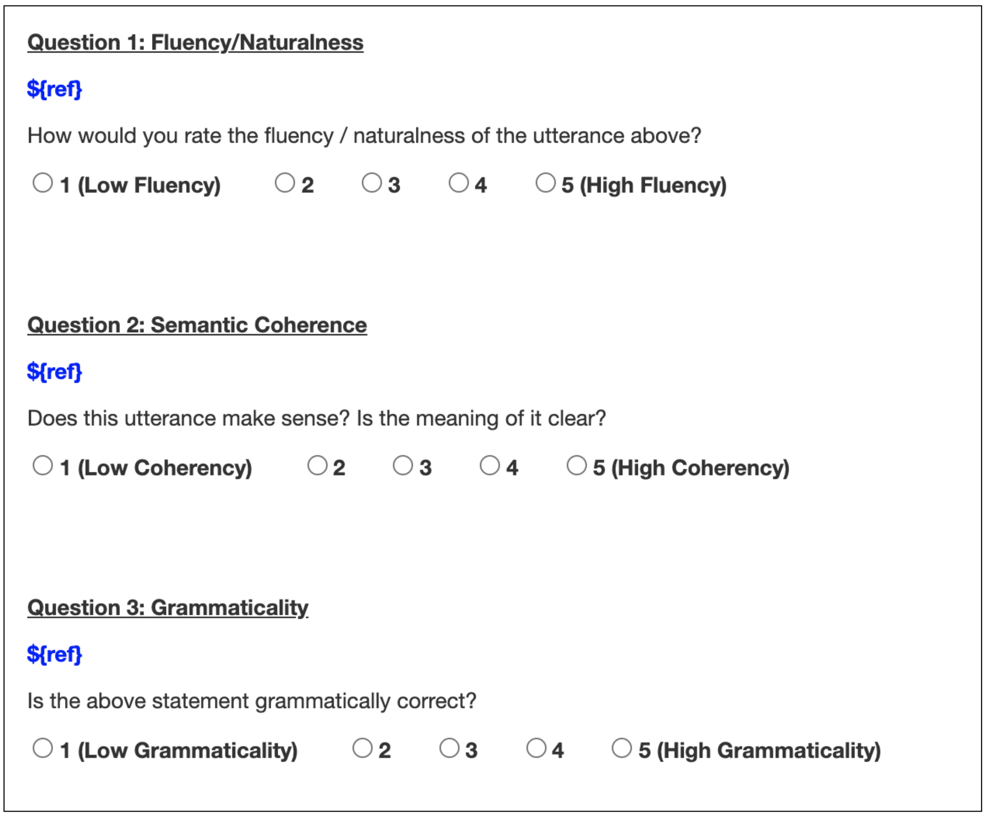
While all of these metrics have allowed for quick evaluation of NLG models, helping prevent a bottleneck caused by human evaluation, they are not without their issues. Essentially, since the release of BLEU, there have been studies which have pointed out and analyzed their flaws, starting as early as 2 years after BLEU was published [Zhang et al., 2004]. Specifically in the area of variations in text there have been studies which show that these metrics do not correlate highly with human evaluation of either fluency or meaning equivalence [Stent et al., 2005]. This finding is important to this work because we are also attempting to generate outputs that are different from the training data to establish how models generalize. These results have not prevented BLEU and the other metrics from becoming the standard of NLG metrics, and in fact the number of papers citing these metrics have grown almost every year in the past two decades regardless of the work being published pushing back against them [Sai et al., 2020]. Even so, there have been many recent papers that continue to demonstrate that these evaluation metrics are insufficient for many tasks and need to be improved or replaced with alternative metrics [Sulem et al., 2018; Sellam et al., 2020; Mathur et al., 2020]. For these reasons, the evaluation metrics that we focused on in this paper were not the automatic evaluation metrics like BLEU. They were instead automatic metrics that we created and designed. We used automatic metrics to evaluate both the semantic accuracy of the output and the stylistic features that we are generating. We also used human evaluation where appropriate, as it is still considered the gold standard way to evaluate aspects of text such as fluency.

## 4.3 Human Evaluation

Human evaluation, while slow and expensive, is still necessary to judge some qualities in text that we currently do not have automatic metrics for. In the experiments, we use human evaluation to measure the naturalness/fluency, grammaticality and semantic correctness of the outputs. For some experiments, we also ask for evaluations on the personality of the output. While the automatic evaluation metrics like BLEU compare the results to reference text, they do not explicitly measure any of these features. Therefore, in order to get an accurate representation of these qualities, we need to ask annotators to evaluate the output. We did not evaluate the results of all models with human annotators, only the models which do best according to the other metrics.

We perform human evaluation using Mechanical Turk. We restrict the annotators to those who have 98% accuracy on other experiments and who have done at least 1000 other HITs. We do this to verify that the annotators are experienced and will answer the questions seriously. We also limit the locations of the Turkers to the USA, Canada, the UK, Australia and New Zealand to confirm that the Turkers are proficient in English. This is necessary because, when asking about qualities like naturalness it is important that the annotators are as fluent as possible in English. For each HIT we have five annotators rate the utterance so that we can calculate agreement.

The HITs are designed to first describe the task that the annotators are going to perform. We then give them the utterance that they are annotating. We define fluency/naturalness by asking "Does the utterance sound fluent and natural? As though it could be said by a human?". Semantic coherency is defined as "Does it make sense? Is the meaning clear?". Finally, when we ask about grammatically, we ask "Is the utterance grammatically correct? Does it follow the rules of English grammar?". We then ask the five annotators to rate how well the utterance fulfills each category on a 5 point Likert scale, where 1 means that the utterance does not fulfill the category at all, and 5 means that the category is perfect in the utterance. An example of a HIT is in Figure 4.1.



**Question 1: Fluency/Naturalness**

**${ref}**

How would you rate the fluency / naturalness of the utterance above?

○ **1 (Low Fluency)** ○ **2** ○ **3** ○ **4** ○ **5 (High Fluency)**

**Question 2: Semantic Coherence**

**${ref}**

Does this utterance make sense? Is the meaning of it clear?

○ **1 (Low Coherency)** ○ **2** ○ **3** ○ **4** ○ **5 (High Coherency)**

**Question 3: Grammaticality**

**${ref}**

Is the above statement grammatically correct?

○ **1 (Low Grammaticality)** ○ **2** ○ **3** ○ **4** ○ **5 (High Grammaticality)**

Figure 4.1: An example of the questions asked in a Mechanical Turk HIT.

We also have to determine if the annotators are in agreement about their ratings to determine how reliable the annotations are. If the annotators have very different labels for each category, then the results are less reliable, since the results are quite different from person to person, so it is either hard to tell how well the utterance does on the output, or the annotators themselves were unreliable. We calculate average Pearson correlation across the five annotators to determine agreement.

While we do sometimes ask the annotators to rate semantic correctness, we do not give them the meaning representation (MR) nor do we ask them to compare the MR to the output. This is because we have an automatic metric for semantic accuracy in reference to the MR. We will go over this metric in the following section (Section 4.4).

## 4.4 Slot Error Rate (SER)

As mentioned previously, slot error rate, or SER, is a calculation of the correctness of the semantics of an output from a data-to-text model, based on the input data. Word error rate (WER) is a common metric that is the inspiration behind SER. WER is commonly used in speech recognition and machine translation and is derived from the Levenshtein distance. Speech recognition and machine translation, in general, have an exact "correct" response for the output of their systems. Therefore, one can measure the performance of these models by checking the accuracy of the exact words. In other NLG problems, this is not possible as there are often many different potential correct outputs. What we do have in data-to-text generation is the desired semantics of the output. There is an expectation that the model will generate everything in the MR,

| **Error Type** | **MR** | **Realization** |
|---|---|---|
| Delete CUISINE, PRICE, NEAR | name[RESTAURANT] cuisine[mexicain] location[midtown] price[expensive] eatType[coffee_shop] familyFriendly[no] near[point-of-interest] | [RESTAURANT] is a coffee shop that is not family friendly. It is located in Midtown. |
| Repeat LOCATION, DECOR | name[RESTAURANT] decor[good] location[midtown_west] eatType[coffee_shop] rating[1_out_of_5] | [RESTAURANT] is a coffee shop in Midtown West with good ambiance. It is in Midtown West with good decor. |
| Substitution QUALITY BAD to GOOD | name[RESTAURANT] decor[good] qual[bad] location[tribeca/soho] eatType[pub] | [RESTAURANT] is in Tribeca/Soho with good food and good decor. It is a pub. |
| Hallucination SERVICE | name[RESTAURANT] decor[good] qual[good] location[riverside] near[point-of-interest] | [RESTAURANT] is near [POINT-OF-INTEREST] in the riverside area. It has good food, good decor and good service. |

Table 4.1: Model outputs illustrating types of semantic errors.

without hallucinations, deletions, substitutions or repetitions. Considering these are the same types of errors the WER uses, we decided to also calculate semantic accuracy by taking the number of errors and normalizing them by the number of semantic features that were expected in the output. The number of features is the number of attributes or slots, and therefore we are calculating the *slot* error rate.

In order to calculate the SER, we define four types of errors that can occur when realizing the slots in the MR. The first is deletion, which is when an attribute that is in the MR is not in the realization. The second is substitution, which is where an attribute in the MR is realized but the wrong value is used. For example, the MR contains the attribute-value pair "cuisine[French]", but the output utterance instead says "serves Spanish food". The attribute defining the cuisine being served in the restaurant is present in the utterance, but the type of food, the value, is incorrect. The third type of error is hallucination, which is where an attribute is not in the input MR but it is in the realization. And finally, the fourth type of error is a repetition. This is where an attribute is mentioned more than once in the output realization. Repetitions and co-occur with the other errors, so one attribute can be at the same time a substitution and a repetition. Repetitions are also defined at the attribute level. If the same attribute is repeated with two different values, it is still considered a repetition and is often also a contradiction, such as saying the restaurant has both good and bad service. The values are different, but this is not a correct utterance because both statements cannot be true simultaneously. Examples of all four errors are in Table 4.1.

We then define SER as:

$$SER = \frac{D + S + H + R}{N}$$

where $D$ is the number of deletions, $S$ is the number of substitutions, $H$ is the number of hallucinations, $R$ is the number of repetitions and $N$ is the number of slots in the input MR. The SER is calculated for each output and then averaged over the entire test set. Since fewer errors is better, the best SER value is zero and a higher SER indicates poor performance. Also, the SER can be higher than one since the number of errors is not limited to the number of slots.

We are able to automatically calculate the SER by taking advantage of the fact that the datasets are synthetic and crowdsourced. Both datasets do not have an unlimited way to realize the values in the MR and we therefore are able to generalize the potential ways the attributes will appear in the text. By searching for potential keywords, key phrases or combinations thereof, we are able to approximate which attributes and values are in the utterance using scripts. We verified that these estimates are fairly accurate, which we will go into more depth on later. First, we will go over how we evaluate which attributes and values are in the dataset for both the NYC and the E2E datasets.

Synthetic datasets have a limited number of ways to generate the desired output for each attribute. For the NYC dataset we were able to take advantage of this fact to determine exactly which attributes and which values were in the text. There are three types of attributes in the NYC dataset, the DA attribute *recommendation*, the attributes where the value describes the polarity of the attribute (*food_quality, service, decor, price*) and the attributes where the value is a direct description of the restaurant (*cuisine, location*). The crowdsourced dataset has more variability in how the attributes are realized, but, because the crowdworkers were given similar queues for the same attributes and values they also have some patterns that we found and used to automatically evaluate the utterances. Most of the attributes in E2E are categorical, except for *family_friendly*, which is a boolean value, which we handle differently. For both NYC and E2E, since the attribute *name* is in every utterance, it is the subject of the utterance and is in both NYC and E2E we do not include it in the SER calculation. There are no utterances that miss it, it does not have a value since it is always delexicalized, and it can accurately occur multiple times if there are multiple sentences considering it is a valid option to repeat the subject instead of using pronouns for later sentences.

The attributes unique to NYC can all be evaluated using set lists of potential values given the input to the MR. For the DA attribute, which in NYC is only *recommendation*, there are 12 ways to represent this feature for each of the two values, a positive recommendation or a negative recommendation, so we search the text for all of these values, if one is present we know the attribute is represented in the output. For the three attributes where the value describes the polarity which are unique to NYC, (*food_quality, service, decor*), each polarity value, terrible, bad, decent, good and terrific, have multiple potential values which can represent these values, such as for *food_quality*, terrific can be represented with adjectives such as "terrific", "amazing", "superb", "delicious" and more. Also, some values can be shared across attributes, so "amazing" can be used to not only describe *food_quality* but also *service* and *decor*. To make sure we have found the correct attribute we also analyze the noun these descriptors are attached to, so *food_quality* is represented using "food", *service* is represented using "service" and "staff" and *decor* is represented using "decor", "atmosphere" and "ambiance". Finally, because some adjectives can be used for multiple polarities, "bad" for example is often used to represent both "bad" and "terrible", we assume the model is correct if possible, so if the MR has "service[terrible]" and *service* is represented in the output as "the service was bad", we assume that the intended value was terrible. We do this for all attributes that have potential overlap in realizations.

For E2E we have two types of datasets, synthetic and crowdsourced. For the synthetic dataset, and for the categorical attributes in the NYC dataset, we can search the text for the values in the attribute. For *location*, *cuisine*, *eat_type*, *customer_rating*, *price* and *near* we are looking for the exact value in the attribute because when PERSONAGE generates the training data it inserts the exact value into the text, so if the location is riverside, PERSONAGE will always include "riverside" in the text. Therefore the only way for the model to learn to correctly generate these attributes is by realizing the exact values in the MR. For the boolean attribute, *family_friendly*, we need to check if there is a negation preceding the attribute, so whether the text is "is family friendly" or "isn't family friendly". There are multiple ways this can be realized, so we check for them all.

When the E2E dataset is crowdsource there are more potential ways to generate these attributes. For example, for the attributes customer rating with the value 1 out of 5 the crowdworker could have included this attribute with the following phrases: "1 out of 5 stars", "1 star", "low rating", etc. In order to correctly analyze the output we search for all these options, collecting potential phrases by manually going through the training data and finding the most common ways to represent each value. Again, since customer rating can also have the value low, "low rating" can be used to represent both so we prioritize the value in the original MR.

In order to confirm that the SER calculations were accurately capturing the semantic accuracy in the outputs we evaluated it with an experiment where two NLG experts hand-labelled a random sample of 200 model outputs. Over the 200 samples, the automatic SER was 0.45 and the human was 0.46. The overall correlation of the automatic SER with the human SER over all types of errors (deletions, repetitions, substitutions and hallucinations) is 0.80 and the correlation with deletions, the most frequent error type, is 0.97.

## 4.5 Stylistic Evaluation

We have now gone over how we automatically evaluated the semantics of the outputs, but these are not the only desired features of the utterances that we can automatically evaluated. Stylistic features are important in NNLG to generate interesting and varied output. If the models are always generating the same attributes and values with the same word choice, in the same order, we might as well just use templates. In the experiments, we both evaluate the models for overall stylistic variety as well as controlled style, where there is an expected stylistic feature and we must confirm that the model properly generated it.

For measuring overall style we measure Shannon text entropy [Shannon, 2001], which quantifies the amount of variation in the output of the models. We calculate it using the following formula:

$$-\sum_{x \in S} \frac{freq}{total} * log_2(\frac{freq}{total})$$

where $S$ is the set of unique words in all outputs generated by the model, $freq$ is the frequency of a term, and $total$ counts the number of terms in all references. When we test entropy we do so for multiple n-grams to determine if the output not only has variety in words, but also combinations of words. The larger the entropy the more stylistically varied the text. $\frac{freq}{total}$ represents the probability that any word in the references is the term $x$, and $-log_2(\frac{freq}{total})$ is the amount of surprisal (how unexpected the term is) for $x$, where low probability terms have high surprisal, so entropy is the average amount of surprise when a term occurs. The more unexpected a term is, the more variety in the output.

Some of the experiments also have specific desired stylistic features in the output, and it is important that the models accurately realize these features. Being able to automatically evaluate the quality and accuracy of these models based on these features is important to allow us to quickly iterate through many models, just as it is for semantic evaluation.

We have two main ways the model tries to generate stylistic features. The first is, given a category label, we want the model to only generate features within that category. The other type of stylistic generation is individual feature generation. We generally have a feature attribute and a label, for example the feature of sentence aggregation and the value of number of sentences, where we need to confirm both the presence of the feature as well as the accuracy of the feature. We will go over how we evaluate both types of stylistic feature generation here.

For categorical feature labels we are evaluating both the pragmatic markers and the aggregation operations which make up the category. We then determine which features are in each utterance in the output and take the Pearson's correlation between the output of the model and the reference text. While these metrics give good estimates on the overall correctness of the models, they do not show the fine-grained accuracy on different features, so we also generate graphical representations to compare the different models in a more interpretable manner.

For individual feature generation we determine if both the feature is present, and in some cases, if the feature is accurately generated. Each feature in these experiments has a value associated with it. For example, when controlling the number of sentences in the generated output, we have to not just determine that there are sentences in the output, but that the value matches. We take both the accuracy of these experiments, but also in some cases the correlation. We also determine if the feature is correctly generated, such as in the case with generating contrast, we need to determine that contrast is present, but also that the attributes being generated are able to be contrasted.

## 4.6 Summary

Evaluating the results is an integral part of Neural Natural Language Generation. Because we are able to train models fairly quickly, automatic evaluation is paramount to iterating through multiple models and performing learning techniques such as self-training. In this chapter we explained why existing automatic evaluation metrics are not sufficient for the tasks that we are experimenting on. Because of this we introduce multiple metrics that use the data-to-text structure of the experiments, and that we want specific features in the outputs, to automatically evaluate both the semantics and stylistic features of the outputs. We also explained and presented the human evaluation procedure, since some features are not currently able to be accurately evaluated automatically, such as fluency, and therefore human evaluation is still necessary. These metrics will be used in the experiments described in chapters 5, 6 and 7.

# Chapter 5

# Personality Control and Generalization

## 5.1 Overview

At this point in the thesis we have defined why generalization is needed in Neural Natural Language Generation (NNLG) and introduced the datasets and evaluation metrics that we will use in the experiments in generalization. In Chapter 3 we described the sources of the datasets we will use in this and the following experiments and in Chapter 4, we demonstrate that the standard automatic evaluation metrics are insufficient and define new ones to evaluate the experiments. In this chapter we go over the first experiments in generalization where we test neural models' architectures to demonstrate what these systems can achieve in terms of controlling and generalizing the stylistic features of the output text [Oraby et al., 2018b,a].

We used the Statistical Natural Language Generator (SNLG) PERSONAGE [Mairesse and Walker, 2007], to generate the training data for these experiments. We went over how PERSONAGE generates data in more detail in Section 3.2.1. Using this generator, we are able to generate not only the semantics required to train the NNLGs, but we are also able to generate controlled stylistic features. PERSONAGE was created with the ability to not only generate individual specific features, but also with combinations of them designed to imitate the Big Five personality traits [Mairesse and Walker, 2010]. These rules govern decisions such as word choice, pragmatic marker insertion and aggregation operations. By generating multiple stylistic features to form a single personality based on pre-defined rules we can learn not only if these models can disentangle style from semantics but also if the model can learn combinations of stylistic features which form abstract concepts such as personality.

In Section 5.2.1, we describe the corpus as well as the personalities that we use in these experiments. We also go into detail about the models we will use. We experiment with three types of supervision. The first is no supervision, which is the baseline to determine what the model can learn if trained with multiple personalities and many stylistic features. The second is a token supervision, where the model is expected to learn the rules associated with a single personality on its own. The final supervision is trained with the personality as well as the features which appear in the utterance, which is the closest to the supervision provided to the SNLG model PERSONAGE.

After training the NNLG's on the models described briefly above we go over the results in Section 5.2.2. We have multiple quantitative and qualitative evaluation metrics. Even though in Section 4.2 we describe why the standard automatic metrics are not very informative we include them for completeness. We also present the slot error rate (SER) results (a metric defined in Section 4.4), and evaluate the stylistic features which make up each personality, and collect human evaluations on the naturalness and perceived personality.

While for the ease of writing we refer to the sets of rules determining the style of the output as personalities, they are actually personality traits. Human personalities can be described as combinations of these traits, so a person can be extroverted and either agreeable or disagreeable. The degree of each of these traits also can vary from person to person. This inspired us to test the model's ability to generalize from training data which has separate personality traits and create novel output that has multiple personality traits, which we refer to as multivoice. To do this we create a new test set where we signal the model to generate two personalities, instead of just one personality, while still using a model trained on data with single personalities.

We go over the experimental framework for this multivoice experiment in Section 5.3.1, then in Section 5.3.2 we describe the semantic and stylistic evaluation of this output where for each multivoice pair we compare the results to both single voices and show that the results are not just a simple average of the two personalities but are a novel voice.

## 5.2 Personality Generation

As a first step toward generalizing the output of trained NNLG models, we are learning general representations of multiple stylistic features. While NNLG models are capable of controlling stylistic features when generating text, generating with overarching style labels which encompass multiple features, such as polite, allows for variation in generation without having to hand tune many features. Here we performed a controlled experiment where we had five general labels for sets of features grounded in psychological context, so stylistic features which one would expect to see together. We test whether the model can learn to generate the style, while preserving semantics with only the general label as supervision, or if the model requires more specific supervision.

These five general labels are from the Big Five personality traits, where each trait is defined by many content planning schemas, aggregation operations and pragmatic markers which are grounding in psychological literature [Mairesse and Walker, 2010]. We selected the personalities agreeable, disagreeable, conscientious, unconscientious, and extrovert, all of which have distinct rules, though there is some overlap. We generated the data using PERSONAGE (Section 3.2.1) and meaning representations (MRs) from the E2E Generation Challenge (Section 3.3) and creating a synthetic E2E dataset (Section 3.4. We use the metrics defined in Chapter 4 to automatically evaluate the outputs of the models and then perform human evaluation on the best model. We experimented with controlling stylistic variation using NNLG by creating references with different personalities and testing if the models could learn the pragmatic markers and aggregation operations associated with the personalities [Oraby et al., 2018b].

### 5.2.1 Experimental Framework

We hypothesized that we could create an NNLG model that could learn the different pragmatic markers and aggregation operations that form the personalities in PERSONAGE and determine whether we can control these features with a single, general, personality token. With this in mind, we created a dataset with data from five personalities from PERSONAGE and trained multiple models with increasing levels of supervision to learn which could generate the stylistic variation we desired while maintaining semantic fidelity. We will go over details about generating this personality dataset and the model architecture in this section.

#### 5.2.1.1 Corpus Creation

For this experiment, we created a new data-to-text dataset of restaurant reviews. This is a synthetic dataset with MRs from the E2E Generation Challenge [Novikova et al., 2017b] that is generated using PERSONAGE [Mairesse and Walker, 2010]. We created a deep syntactic structure representation for each of the eight attributes in the E2E MRs, which PERSONAGE uses to generate references. As mentioned previously, there are more details about this process in Chapter 3 where we describe both how PERSONAGE works and the E2E dataset. For this corpus, we generated the datasets with personalities informed by the Big Five Personality Traits and the characteristics defined by the high and low extremes for each personality trait. In Table 5.1, we present these personalities and their high and low score labels, which from now on we will refer to as the "personalities". For each reference, we generated the references with one of

| Personality Trait | High Score Label | Low Score Label |
|---|---|---|
| EXTROVERSION | **extrovert** | introvert |
| AGREEABLENESS | **agreeable** | **disagreeable** |
| OPENNESS | open | not open |
| CONSCIENTIOUSNESS | **conscientious** | **unconscientious** |
| EMOTIONAL STABILITY | emotionally stable | neurotic |

Table 5.1: The five personalities of the Big Five Personality Traits and their high and low score labels. The trait labels that we experimented with are in red.

five personalities: agreeable, disagreeable, conscientious, unconscientious, and extrovert, which are marked in red in the table. We are able create many different outputs for a single MR by using the different pragmatic markers and aggregation operations that each personality can use and the content and sentence planning that PERSONAGE uses to generate different variations. By generating synthetic data with PERSONAGE we are able to have a controlled dataset with specific stylistic features associated with each personality. This allows us to test how different neural architectures affect the ability of the trained model to disentangle style from content and faithfully produce semantically correct utterances that vary style.

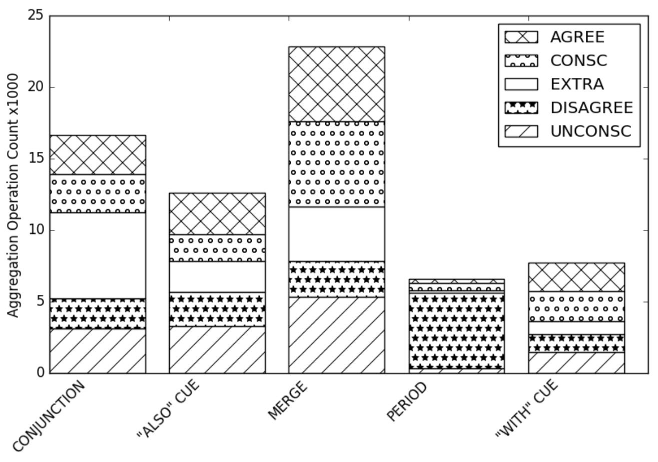
As mentioned previously, the dataset consists of five personalities, agreeable, disagreeable, conscientious, unconscientious and extrovert. We selected these five because they have easily visible and distinct traits and they are personalities we could conceivably want to generate reviews using. Extroversion, as the most important of the five traits and a personality which attracts others, was especially important to include [Goldberg, 1990]. Agreeable and conscientious could make positive reviews appear even more positive and disagreeable and unconscientious would emphasize the negativity in poor reviews, so we also select these personalities, though for this experiment we do not define reviews as overall positive and negative and therefore do not limit a personality to certain attribute and value pairs. We do not include introvert because being shy is not a desirable trait for restaurant reviews; neurotic and emotionally stable are also not included for a similar reason. Openness to experience and not open are the least visible and weakest among the Big Five personality traits, so we also do not include those personalities [Peabody and Goldberg, 1989]. We wanted a stylistically varied corpus and one of the main distinctions between the different personality models is their use of aggregation operations and pragmatic markers. In Table 5.2, we show the aggregation operations and pragmatic markers used and their level of intensity in the five personalities, examples of these features can be seen in Table 3.1. Each personality has a distinct set of features though not all features are unique to a single personality. Agreeable and conscientious, for example, both have high levels of confirmation requests and general softeners, though these are not the same personality regardless of this overlap, as other features are only included for one of these personalities. Figure 5.1 shows the proportion of all the aggregation operations and a sample of the pragmatic markers, from Table 3.1, for the different personalities. This demonstrates that although some traits are high for multiple personalities, this does not mean they appear in the same rate for both personalities. So even though general softeners is high for both agreeable and conscientious you can see they do not appear in both personalities to the same degree. This shows that each personality produces stylistically distinct data.

We used the train, development and test split used in the E2E Generation

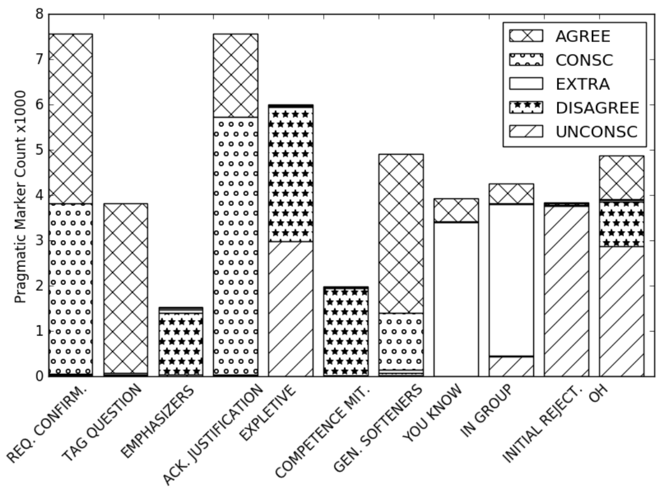
| **Attribute** | **Agree** | **Disag** | **Consc** | **Uncon** | **Extr** |
|---|---|---|---|---|---|
| Aggregation Operations | | | | | |
| Period | low | high | mid | mid | low |
| “With” cue | mid | mid | mid | mid | low |
| Conjunction | high | low | mid | mid | high |
| All Merge | mid | mid | mid | mid | mid |
| “Also” cue | high | low | mid | mid | high |
| Ellipsis | high | low | mid | mid | high |
| Pragmatic Markers | | | | | |
| acknowledge_definitive | high | low | mid | mid | low |
| acknowledge_justification | high | low | high | low | low |
| acknowledge_yeah | high | low | low | high | high |
| down_kind_of | high | low | low | high | high |
| down_like | mid | mid | low | high | high |
| down_around | high | low | mid | mid | mid |
| down_subord | high | low | high | low | low |
| general softener | high | low | high | low | low |
| emphasizer | low | high | mid | mid | high |
| emph_you_know | high | low | mid | mid | low |
| exclaim | mid | mid | low | high | high |
| expletives | low | high | low | high | mid |
| near expletives | mid | mid | low | high | high |
| indicate surprise | mid | mid | mid | mid | mid |
| initial rejection | low | high | low | high | mid |
| competence mitigation | low | high | mid | mid | mid |
| filled pause stative | mid | mid | low | high | low |
| confirmation request | high | low | high | low | high |
| in group marker | high | low | low | high | high |
| tag question | high | low | mid | mid | high |

Table 5.2: Aggregation and pragmatic markers produced by different personalities.

Challenge, since the three sets had unique MRs. Since did not need a development set we used both the train and development MRs in the training dataset. We had 3,784 unique MRs in the training set and generated 17,771 utterances per personality for a training set of 88,855 utterances. The test set consists of 278 unique MRs and we generate five utterances, one per personality, for a test set of 1,390 utterances. Examples of



(a) Aggregation Operations.



(b) Pragmatic Markers.

Figure 5.1: Frequency of the most frequent Aggregation and Pragmatic Markers in the Training Data.

the utterances can be seen in Table 5.3. Each personality has a different distribution of pragmatic markers and aggregation operations, as mentioned previously and can be seen in these examples. For example, both the AGREEABLE and CONSCIENTIOUS personalities use a *confirmation request*, such as “Let’s see what we can find...”, but AGREEABLE also has the *tag question* “you see?” which CONSCIENTIOUS does not. UNCONSCIEN-

TIOUS uses *expletives* and *initial rejection*s, like "I don't know" and EXTROVERT uses emphasizers, such as "basically". They also have different aggregation operations, with DISAGREEABLE having many, smaller, sentences, while for the rest of the personalities all of the attributes are in a single sentence. Also, AGREEABLE uses the *also* cue twice and UNCONSCIENTIOUS uses it once. The variation in these personalities is clear with just a single reference for each one. These personalities have lots of stylistic variation that the model will need to learn.

#### 5.2.1.2 Model Architectures

The neural generation models build on the open-source sequence-to-sequence (seq2seq) TGen system [Dusek and Jurcícek, 2016a], implemented in Tensorflow [Abadi et al., 2016]. The system is based on seq2seq generation with attention [Bahdanau et al., 2014; Sutskever et al., 2014] and uses a sequence of LSTMs [Hochreiter and Schmidhuber, 1997] for the encoder and decoder, combined with beam-search and reranking for output tuning.

The inputs to TGen are dialogue acts for each system action (such as *inform*) and a set of attribute slots (such as *rating*) and their values (such as *high* for attribute *rating*). The system integrates sentence planning and surface realization into a single step to produce natural language outputs. To preprocess the corpus of MR/utterance pairs, attributes that take on proper-noun values are delexicalized during training i.e., *name* an d *near*. During the generation phase on the test set, a post-processing step relexicalizes the outputs. The MRs (and the resultant embeddings) are sorted internally

| **Personalities** | **Realization** |
|---|---|
| MEANING REPRESENTATION (MR) | NAME[FITZBILLIES], EATTYPE[PUB], FOOD[ITALIAN], PRICERANGE[MODERATE], CUSTOMER RATING[DECENT], AREA[RIVERSIDE], FAMILYFRIENDLY[NO],NEAR['THE SORRENTO'] |
| NO-AGG/ NO-PRAG | Fitzbillies is a pub. Fitzbillies has a decent rating. Fitzbillies is moderately priced. Fitzbillies is in riverside. Fitzbillies is an Italian restaurant. Fitzbillies is not family friendly. Fitzbillies is near The Sorrento. |
| AGREEABLE | **Let's see what we can find on Fitzbillies. I see, well** it is a pub with a decent rating, also it is an Italian restaurant in riverside and moderately priced near The Sorrento, also it isn't family friendly, **you see?** |
| DISAGREEABLE | **I mean, everybody knows that** moderately priced Fitzbillies is in riverside with a decent rating. It's near The Sorrento. It isn't family friendly. It is an Italian place. It is a pub. |
| CONSCIENTIOUS | **Let's see what we can find on Fitzbillies. I see, well** it is a pub with a decent rating, it isn't kid friendly and it's moderately priced near The Sorrento and an Italian restaurant in riverside. |
| UNCONSCIENTIOUS | **Oh god yeah, I don't know.** Fitzbillies is a pub with a decent rating, also it is moderately priced near The Sorrento and an Italian place in riverside and it isn't kid friendly. |
| EXTROVERT | **Basically,** Fitzbillies is an Italian place near The Sorrento and actually moderately priced in riverside, it has a decent rating, it isn't kid friendly and it's a pub, **you know.** |

Table 5.3: Sample neural model output realizations for the same MR for PERSONAGE's personalities, Pragmatic markers in the utterances are in bold.

by dialogue act tag and attribute name.

The models are designed to systematically test the effects of increasing the level of supervision, with novel architectural additions to accommodate these changes. We use the default parameter settings from TGen [Dusek and Jurcícek, 2016a] with batch size 20 and beam size 10, and use 2,000 training instances for parameter tuning to set the number of training epochs and learning rate. Figure 5.2 summarizes the architectures.

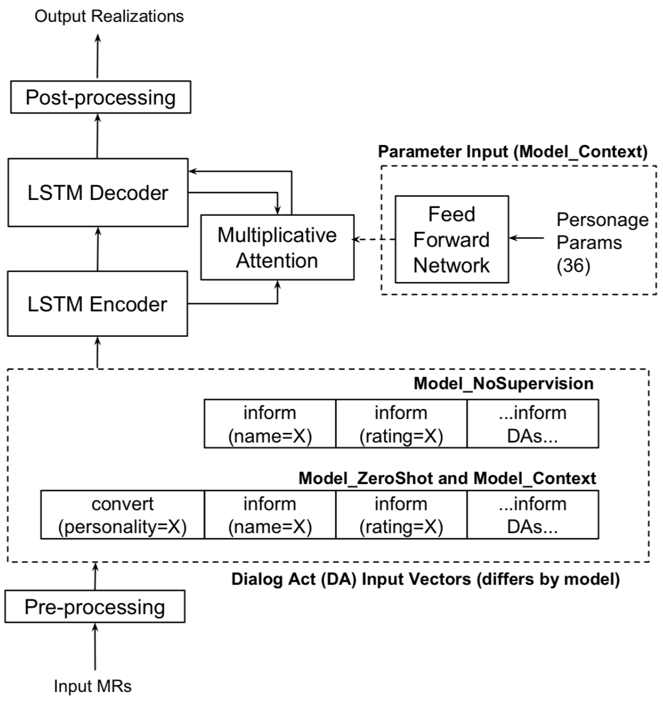


Figure 5.2: Neural network model architectures for different levels of supervision to represent personality.

As mentioned previously, we experimented with three levels of supervision and tuned the parameters of the model for each type of supervision separately.

The simplest model, which has no additional supervision and we will refer to as MODEL_NOSUP from now on, follows the baseline TGen architecture [Dusek and Jurcícek, 2016b], with training using all 88K utterances in a single pool for up to 14 epochs based on loss monitoring for the decoder and reranker.

The second model adds a token of additional supervision by introducing a new dialogue act, *convert*, to encode personality, inspired by the use of a language token for machine translation [Johnson et al., 2017]. Unlike other work that uses a single token to control generator output [Fan et al., 2018; Hu et al., 2017], the personality token encodes multiple different parameters that define the style of the matching reference, generalizing this many parameters into a single label. Uniquely here, the model attempts to *simultaneously* control multiple style variables that may interact in different ways. This model will be referred to as MODEL_TOKEN. Again, we monitor loss on the validation set and training continues for up to 14 epochs for the decoder and reranker.

The most complex model introduces a context vector, as shown at the top right of Figure 5.2. The vector explicitly encodes a set of 36 style parameters from Table 5.4. The parameters for each reference text are encoded as a boolean vector where if the parameter is present in the text the style parameter is set to True, otherwise it is set to False. A feed-forward network is added as a context encoder, taking the vector as input to the hidden state of the encoder and making the parameters available at every time step to a multiplicative attention unit. The activations of the fully connected nodes are represented as an additional time step of the encoder of the seq2seq architecture. The attention is computed over all of the encoder states and the hidden state of the fully connected network. Again, we set the learning rate, alpha decay, and maximum training epochs (up to 20) based on loss monitoring on the validation set.

### 5.2.2 Results

Here, we present results on controlling stylistic variation while maintaining semantic fidelity.

| | | | |
|---|---|---|---|
| 1: emph_you_know | 10: down_quite | 19: ack_well | 28: init_rejection |
| 2: emph_really | 11: down_rather | 20: ack_oh | 29: tag_question |
| 3: emph_basically | 12: down_around | 21: ack_right | 30: req_confirmation |
| 4: emph_actually | 13: down_like | 22: ack_ok | 31: aggreg_conjunc |
| 5: emph_just | 14: filled_err | 23: ack_i_see | 32: aggreg_ellipses |
| 6: emph_exclaim | 15: filled_mmhm | 24: expletives | 33: aggreg_also |
| 7: down_kind_of | 16: down_subord | 25: near_expletives | 34: aggreg_merge |
| 8: down_sort_of | 17: filled_i_mean | 26: comp_mitigation | 35: aggreg_period |
| 9: down_somewhat | 18: ack_yeah | 27: in_group_marker | 36: aggreg_with |

Table 5.4: The 36 parameters encoded into the context vector used in MODEL_CONTEXT.

#### 5.2.2.1 Evaluating Semantic Quality

First we present the classic automatic metrics used in NLG to measure how well the output compared to the original PERSONAGE input, then we will introduce the novel metrics. The classic evaluation uses the E2E generation challenge script, the summary of the results for automatic metrics are in Table 5.5. Although the results are similar, MODEL_CONTEXT is slightly better than the others.

| **Model** | **BLEU ↑** | **NIST ↑** | **METEOR ↑** | **ROUGE_L ↑** |
|---|---|---|---|---|
| NoSup | 0.2774 | 4.2859 | 0.3488 | 0.4567 |
| Token | 0.3464 | 4.9285 | 0.3648 | 0.5016 |
| Context | **0.3766** | **5.3437** | **0.3964** | **0.5255** |

Table 5.5: Automated Metric Evaluation.

These classic evaluation metrics are not informative about the quality of the references and penalize stylistic variation, so we developed new scripts to automatically evaluate specific features in the text. We went into more detail on these scripts in Chapter 4. Here we use the slot error rate (SER) to evaluate the semantic accuracy of the output. We also evaluate three of the errors that make up the SER separately, deletions, repetitions and substitutions, to see how the model is making errors. Hallucinations did not occur frequently enough to include. The results are in Table 5.6. Note that smaller ratios are preferable since these results represent the number of errors. Also, since MODEL_NOSUP does not encode the personality parameter the results are the same across each personality.

| **Model** | **AGREE ↓** | **CONSC ↓** | **DISAG ↓** | **EXTR ↓** | **UNCON ↓** |
|---|---|---|---|---|---|
| DELETIONS | | | | | |
| NoSup | 0.01 | 0.01 | 0.01 | 0.01 | 0.01 |
| Token | 0.27 | 0.22 | 0.87 | 0.74 | 0.31 |
| Context | 0.08 | 0.01 | 0.14 | 0.08 | 0.01 |
| REPETITIONS | | | | | |
| NoSup | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Token | 0.29 | 0.12 | 0.81 | 0.46 | 0.28 |
| Context | 0.02 | 0.00 | 0.14 | 0.00 | 0.00 |
| SUBSTITUTIONS | | | | | |
| NoSup | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 |
| Token | 0.34 | 0.41 | 0.22 | 0.35 | 0.29 |
| Context | 0.03 | 0.03 | 0.00 | 0.00 | 0.03 |
| **SER** | | | | | |
| NoSup | **0.02** | 0.02 | **0.02** | 0.02 | 0.02 |
| Token | 0.13 | 0.11 | 0.27 | 0.22 | 0.13 |
| Context | **0.02** | **0.01** | 0.04 | **0.01** | **0.00** |

Table 5.6: Deletions, Repetitions, Substitutions and SER by Personality.

We can see that MODEL_NOSUP makes the fewest semantic errors, but we will later show that this is at the cost of stylistic variation. After that, MODEL_CONTEXT makes fewer errors than MODEL_TOKEN, which suggests that having more supervision for the stylistic variation also helps avoid semantic errors. We can also see in the last row, that DISAGREEABLE and EXTROVERT have the most semantic errors while CONSCIENTIOUS has the fewest.

#### 5.2.2.2 Evaluating Stylistic Variation

Here we characterize the fidelity of stylistic variation across different model outputs.

**Entropy.** Shannon text entropy quantifies the amount of variation in the output produced by each model. We calculate entropy as $-\Sigma_{x \in S} \frac{freq}{total} * log_2(\frac{freq}{total})$, where $S$ is the set of unique words in all outputs generated by the model, $freq$ is the frequency of a term, and *total* counts the number of terms in all references. Table 5.7 shows that the training data has the highest entropy, but MODEL_CONTEXT performs the best at preserving the variation seen in the training data. Row NOSUP shows that MODEL_NOSUP makes the fewest semantic errors, but produces the least varied output. MODEL_CONTEXT, informed by the explicit stylistic context encoding, makes comparably few semantic errors, while producing stylistically varied output with high entropy.

| **Model** | **1-grams** | **1-2grams** | **1-3 grams** |
|---|---|---|---|
| PERSONAGE TRAIN | 5.97 | 7.95 | 9.34 |
| NOSUP | 5.38 | 6.90 | 7.87 |
| TOKEN | 5.67 | 7.35 | 8.47 |
| CONTEXT | **5.70** | **7.42** | **8.58** |

Table 5.7: Shannon Text Entropy.

**Pragmatic Marker Usage.** We calculate the model's ability to reproduce pragmatic markers by personality by counting the occurences of each pragmatic marker in the reference, averaging those counts and then computing the Pearson correlation between the averages for the PERSONAGE reference and the averages for the output reference for each model and personality, as seen in Table 5.8. The table shows that MODEL_CONTEXT has the highest correlation for all personalities except AGREEABLE. MODEL_NOSUP, on the other hand, does quite poorly for all the personalities except CONSCIENTIOUS, and is in fact negative for three out of the five personalities. This is most likely because the most common pragmatic marker for CONSCIENTIOUS is *request confirmation*, which, since it is just added to the beginning of the reference, is less complex to introduce to the utterance.

| **Model** | **AGREE ↑** | **CONSC ↑** | **DISAG ↑** | **EXTR ↑** | **UNCON ↑** |
|---|---|---|---|---|---|
| NoSup | 0.05 | 0.59 | -0.07 | -0.06 | -0.11 |
| Token | **0.35** | 0.66 | 0.31 | 0.57 | 0.53 |
| Context | 0.28 | **0.67** | **0.40** | **0.76** | **0.63** |

Table 5.8: Correlations between Personage and models for pragmatic markers in Table 3.1.

**Aggregation Operation Usage.** We use the same methods to calculate the models' ability to reproduce aggregation operations by personality, by counting the occurrences of each aggregation operation in the reference, averaging those counts and then computing the Pearson correlation between the averages for the PERSONAGE reference and the averages for the output reference for each model and personality, as seen in Table 5.9. The table shows that, again, MODEL_CONTEXT has a higher correlation for all the personalities except one, DISAGREEABLE in this instance. Because aggregation operations are necessary to form a reference, unlike pragmatic markers, MODEL_NOSUP actually outperforms MODEL_TOKEN for some personalities such as AGREEABLE, CONSCIENTIOUS and UNCONSCIENTIOUS.

| Model | AGREE ↑ | CONSC ↑ | DISAG ↑ | EXTR ↑ | UNCON ↑ |
|---|---|---|---|---|---|
| NoSup | 0.78 | 0.80 | 0.13 | 0.42 | 0.69 |
| Token | 0.74 | 0.74 | **0.57** | 0.56 | 0.60 |
| Context | **0.83** | **0.83** | 0.55 | **0.66** | **0.70** |

Table 5.9: Correlations between Personage and models for aggregation operations in Table 3.1.

By creating these new tools to evaluate the outputs of NNLG models we are able to quickly and accurately get feedback from the experiments. These experiments were focused on very specific aspects of the reference, which were measurable through specific scripts. While these scripts do not evaluate the naturalness of the data, neither do the classic automatic metrics, and unlike the classic metrics, these scripts are very transparent as to what they are measuring and why.[1]

#### 5.2.2.3 Human Analysis

Based on the quantitative results, we select model_context as the best-performing model and conduct an evaluation to test if humans can distinguish the personalities exhibited. We randomly select a set of 10 unique MRs from the personage training data along with their corresponding reference texts for each personality (50 items in total), and 30 unique MRs model_context output (150 items in total).[2] We construct a HIT on Mechanical Turk, presenting a single output (either personage or model_context, and ask 5 Turkers to label the output using the Ten Item Personality

[1]These results were improved upon in [Harrison et al., 2019] by using an update seq2seq model where the hyper-parameters were tuned using a grid search method over the RNN layers and size

[2]Note that we use fewer personage references simply to validate that the personalities are distinguishable in the training data, but will more rigorously evaluate the model in future work.

Inventory (TIPI) [Gosling et al., 2003]. The TIPI is a ten-item measure of the Big Five personality dimensions, consisting of two items for each of the five dimensions, one that *matches* the dimension, and one that is the *reverse* of it, and a scale that ranges from 1 (disagree strongly) to 7 (agree strongly). To qualify Turkers for the task, we ask that they first complete a TIPI on themselves, to help ensure that they understand it.

Table 5.10 presents results as aggregated counts for the number of times at least 3 out of the 5 Turkers rated the *matching* item for that personality higher than the *reverse* item (Ratio Correct), the average rating the correct item received (range between 1-7), and an average "naturalness" score for the output (also rated 1-7). From the table, we can see that for PERSONAGE training data, all of the personalities have a correct ratio that is higher than 0.5. The MODEL_CONTEXT outputs exhibit the same trend except for UNCONSCIENTIOUS and AGREEABLE, where the correct ratio is only 0.17 and 0.50, respectively (they also have the lowest correct ratio for the original PERSONAGE data).

Table 5.10 also presents results for naturalness for both the reference and generated utterances, showing that both achieve decent scores for naturalness (on a scale of 1-7). While human utterances would probably be judged more natural, it is not at all clear that similar experiments could be done with human generated utterances, where it is difficult to enforce the same amount of experimental control.

|  | PERSONAGE |  |  | MODEL_CONTEXT |  |  |
|---|---|---|---|---|---|---|
| **Person.** | **Ratio Correct** | **Avg. Rating** | **Nat. Rating** | **Ratio Correct** | **Avg. Rating** | **Nat. Rating** |
| Agree | 0.60 | 4.04 | 5.22 | 0.50 | 4.04 | 4.69 |
| Disagr | 0.80 | 4.76 | 4.24 | 0.63 | 4.03 | 4.39 |
| Consc | 1.00 | 5.08 | 5.60 | 0.97 | 5.19 | 5.18 |
| Uncon | 0.70 | 4.34 | 4.36 | 0.17 | 3.31 | 4.58 |
| Extr | 0.90 | 5.34 | 5.22 | 0.80 | 4.76 | 4.61 |

Table 5.10: Percentage of Correct Items and Average Ratings and Naturalness Scores for Each Personality (PERSONAGE vs. MODEL_CONTEXT.

## 5.3 Multivoice Personality Generation

The first experiment with generalization beyond training focused on creating outputs with multiple personalities, while training on data with single, separate personalities [Oraby et al., 2018a].

### 5.3.1 Experimental Framework

For this experiment we used the same dataset that we used for the first personality experiment, described in Section 5.2.1.

For the model we also used the same architecture based on TGen for the personality experiment, described in Section 5.2.1. Since we are combining two personalities we used the supervision described for the MODEL_TOKEN model, instead of the MODEL_NOSUP or MODEL_CONTEXT. MODEL_NOSUP does not have the supervision necessary to combine personalities, and MODEL_CONTEXT has too many parameters and is therefore difficult to combine, even though it performed the best out of the three models. We wanted to see if with just the MODEL_TOKEN framework the model could combine the pragmatic markers and aggregation operations seen in both models to create a new personality.

The model was slightly different from MODEL_TOKEN because it is trained on unsorted input, which allows us to add multiple convert tags to the MR for testing. This also means that the training and testing data is different, the training data has one token of supervision and the testing data has two tokens.

When we generate the test references, we also used the same test MRs, but instead of generating one reference per personality we generate two references per combination of two personalities, since the order of the personality tokens matters. We do not have *true* references for these outputs since we do not know what multivoice should be. We have eight combinations since we do not combine personalities that are exact opposites, such as AGREEABLE and DISAGREEABLE. This results in us generating 4,448 total realizations from 278 MRs.

### 5.3.2 Results

**Evaluating Semantic Quality.**

Again, we still report on the classic automatic metrics for completeness, since those are the standard evaluation of quality for NLG models. We use the automatic evaluation scripts from the E2E Generation Challenge, the results of which are summarized in Table 5.11. Multivoice automatically has a better chance of having higher results because the evaluation is over more examples than for single voice. Each multivoice output is compared to two possible references, one for each of the single voice and then averaged.

| Personality | BLEU | NIST | METEOR | ROUGE_L |
|---|---|---|---|---|
| SINGLE VOICE | 0.35 | 4.93 | 0.36 | .50 |
| MULTIVOICE | 0.42 | 5.64 | 0.36 | .52 |

Table 5.11: Automatic Metric Evaluation.

For the semantic evaluations we again calculated the SER as defined in Section 4.4. Table 5.13 has the results for both the multivoice outputs and the single voice outputs for comparison. Semantic fidelity is worse for the multivoice model across most combinations as it is more difficult to preserve semantic fidelity while trying to achieve multiple types of stylistic variation. For deletions, when comparing single to multiple personalities, there are more deletions for the single personality outputs than the multivoice outputs, though they are proportionate. EXTROVERT and DISAGREEABLE have the most deletions for the single personalities, and the multivoice output with the highest numbers of deletions has either EXTROVERT or DISAGREEABLE as one of the personalities. Though sometimes for deletions the worse personality does not significantly hurt the semantic quality, such as for AGREEABLE + EXTROVERT, there are only slightly more deletions than AGREEABLE, and considerably fewer than EXTROVERT. The number of repetitions for multivoice are closer to the single personality outputs, and actually seem to decrease in a number of cases, and same for hallucinations. This shows that adding an extra token not seen in the training data does not seem to significantly hurt the semantic accuracy of the results.

**Evaluating Stylistic Variation.**

| **Personality** | **Deletions** | **Repetitions** | **Hallucinations** |
|---|---|---|---|
| Agree | 0.27 | 0.29 | 0.34 |
| Consc | 0.22 | 0.12 | 0.41 |
| Extr | 0.74 | 0.46 | 0.35 |
| UnConsc | 0.31 | 0.28 | 0.29 |
| Disagree | 0.87 | 0.81 | 0.22 |
| **Personality Pairs** | | | |
| Agree+Consc | 0.44 | 0.08 | 0.26 |
| Agree+Extr | 0.28 | 0.17 | 0.19 |
| Agree+Unconsc | 0.33 | 0.24 | 0.24 |
| Consc+Disagr | 1.01 | 0.18 | 0.28 |
| Consc+Extr | 0.67 | 0.28 | 0.23 |
| Disagr+Extr | 1.20 | 0.75 | 0.09 |
| Disagr+Unconsc | 1.10 | 0.39 | 0.14 |
| Extr+Unconsc | 1.05 | 0.55 | 0.17 |

Table 5.12: Deletions, Repetitions and Hallucinations for Single Voice and MultiVoice Personality Pair.

| **Personality** | **SER** |
|---|---|
| Agree | 0.13 |
| Consc | 0.11 |
| Extr | 0.22 |
| UnConsc | 0.13 |
| Disagree | 0.27 |

(a) Single-Voice model SER.

| **Personality Pairs** | **SER** |
|---|---|
| Agree+Consc | 0.11 |
| Agree+Extr | 0.09 |
| Agree+Unconsc | 0.12 |
| Consc+Disagr | 0.21 |
| Consc+Extr | 0.17 |
| Disagr+Extr | 0.29 |
| Disagr+Unconsc | 0.23 |
| Extr+Unconsc | 0.25 |

(b) MultiVoice model SER.

Table 5.13: Slot Error Rate by MultiVoice Personality Pair as Compared to Single-Voice models.

We calculate the multivoice model's ability to reproduce aggregation operations of both of its parent single voice models by counting the occurrences of each aggregation operation in the reference, averaging those counts and then computing the Pearson correlation between the averages for the personage reference and the averages for the output reference for each model and personality, as seen in Table 5.15. The third and fourth columns are the correlations between the multivoice output and the single personality output and the last column provides correlations between the original two single personalities. From the last column we can see that some personalities are very similar in terms of aggregation, such as the AGREEABLE and CONSCIENTIOUS personalities, and some are very different, such as DISAGREEABLE and EXTROVERT. As we would expect, models that are similar to begin with are less novel when they are combined, as you can see from AGREEABLE + CONSCIENTIOUS. However other combinations seem to produce completely novel models that use aggregation very differently than either of their single voice source models. For example, combining DISAGREEABLE and UNCONSCIENTIOUS produces a model whose use of aggregation is distinct from either of its source models.

| **P1** | **P2** | **P1+P2 vs. P1** | **P1+P2 vs. P2** | **P1 vs. P2** |
|---|---|---:|---:|---:|
| AGREE | CONSC | 0.74 | 0.76 | 0.74 |
| AGREE | EXTR | 0.70 | 0.31 | 0.44 |
| AGREE | UNCONSC | 0.75 | 0.31 | 0.65 |
| CONSC | DISAGR | 0.36 | 0.65 | 0.01 |
| CONSC | EXTR | 0.51 | 0.31 | 0.44 |
| DISAGR | EXTR | 0.53 | -0.36 | -0.04 |
| DISAGR | UNCONSC | 0.23 | 0.33 | 0.05 |
| EXTR | UNCONSC | 0.20 | 0.43 | 0.47 |

Table 5.15: Correlations between PERSONAGE and multivoice models for aggregation operations in Table 3.1.

Figure 5.3 provides a closer look at particular aggregation operations associated with CONSCIENTIOUS and DISAGREEABLE and plots the differences between the single voice models and the use of these operations in the multivoice models. For example, the *period* operation is very common for DISAGREEABLE and not common for CONSCIENTIOUS, but for the multivoice output the use of *period* seems to interpolate the two values. Though some operations counts appear to interpolate the two original voices, some are unique, so the model does not always interpolate, as you can see from the *conjunction* and *also* operations. These plots clearly show that the multivoice model is a novel personality, yielding a different distribution for aggregation operations than either of its source voice styles.

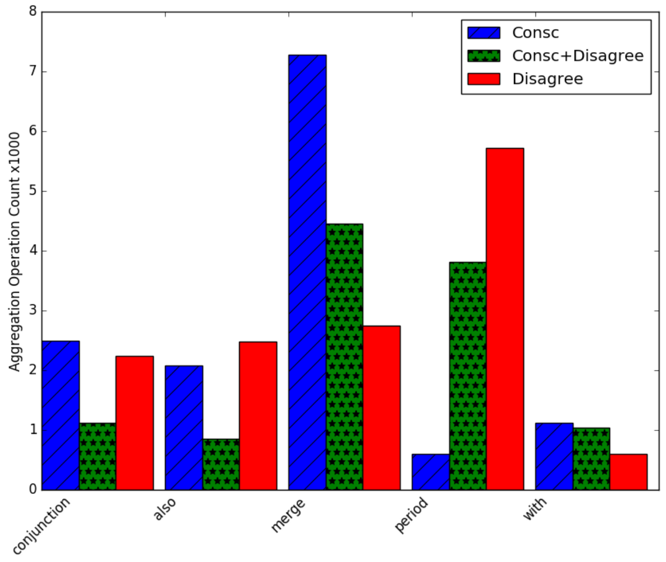


Figure 5.3: Frequency of the most frequent Aggregation Operations for Conscientious and Disagreeable compared to combined Conscientious and Disagreeable Multivoice.

Again, we calculate the multivoice model's ability to reproduce pragmatic markers of both of its parent single voice models by counting the occurrences of each pragmatic marker in the reference, averaging those counts and then computing the Pearson correlation between the averages for the PERSONAGE reference and the averages for the output reference for each model and personality, as seen in Table 5.16. As you can see, while most multivoice models are more similar to one parent model than the other, none are very similar to either of their parents, they each appear to demonstrate characteristics of a novel voice. For example DISAGREEABLE + EXTROVERT bears very little similarity to either DISAGREEABLE or EXTROVERT.

| **P1** | **P2** | **P1+P2 vs. P1** | **P1+P2 vs. P2** | **P1 vs. P2** |
|---|---|---|---|---|
| AGREE | CONSC | 0.11 | 0.74 | 0.30 |
| AGREE | EXTR | 0.19 | -0.02 | -0.07 |
| AGREE | UNCONSC | 0.03 | 0.18 | -0.16 |
| CONSC | DISAGR | 0.44 | 0.05 | -0.10 |
| CONSC | EXTR | 0.41 | -0.09 | -0.11 |
| DISAGR | EXTR | 0.12 | -0.03 | -0.07 |
| DISAGR | UNCONSC | 0.09 | 0.34 | -0.05 |
| EXTR | UNCONSC | -0.11 | 0.37 | -0.08 |

Table 5.16: Correlations between PERSONAGE and multivoice models for pragmatic markers in Table 3.1.

Figure 5.4 provides a closer look at particular pragmatic markers associated with CONSCIENTIOUS and DISAGREEABLE and plots the difference between the single personality models and multivoice models. Most markers are used by only one of the two single voices, while the multivoice model uses most, but not all of them, and the original amount of use does not have a direct correlation to the frequency of use within the multivoice model. The *subord* and *request confirmation* markers are used in very different amounts by the CONSCIENTIOUS voice and similar amounts by the multivoice model. Again, these plots show that the multivoice model is a novel personality that yields a different distribution for pragmatic markers than either of its source voice styles.

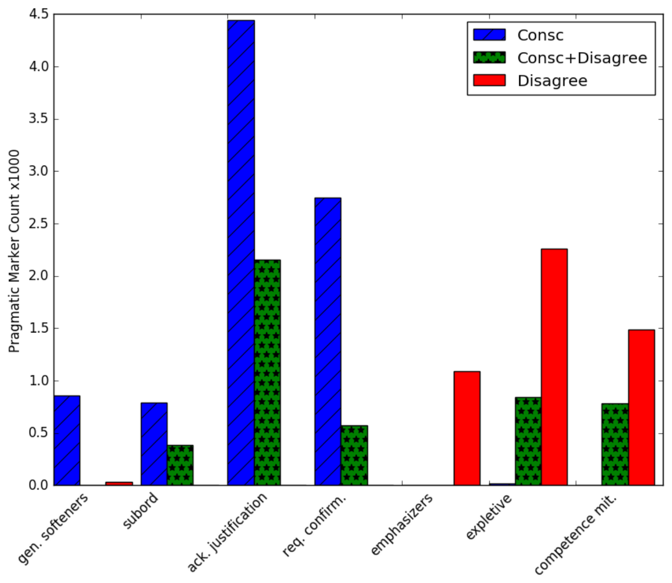


Figure 5.4: Frequency of the most frequent Pragmatic Markers for Conscientious and Disagreeable compared to combined Conscientious and Disagreeable Multivoice.

One limitation is that although we know that the personalities being generated by these multivoice models are novel, we have no way of knowing which personality they actually are because we have no gold standard for multivoice data. But it still shows that we are able to take two separate personalities and combine them to create output that generalizes between the two source personalities.

## 5.4 Summary

In this chapter we went over the first attempts at generalization using NNLG. We first attempted to generalize multiple stylistic features to a single label, using personality to combine different stylistic features in a way that is natural. We found that representing multiple features under a single label results in decent correlation to the original data but degrades the semantic accuracy of results. In Section 5.2, we showed that a more fine-grained representation with each feature represented separately significantly improves the semantic accuracy.

We then further tested the NNLG's ability to generalize by combining personalities together in test that were not seen combined in the training data. In Section 5.3), we used the general token representation from the first experiment and found that testing on two tokens was able to create completely novel combinations of stylistic features. This demonstrates that these models can be extended outside the scope of the original training data and learn to generate without a significant decrease in semantic accuracy.

Next we will describe how we experiment further with these models' ability to learn more than what is seen in the training data and to learn to generate new combinations of stylistic features with new semantics. Chapter 6) presents experiments on learning to generate aggregation on unseen values and improve contrast results by combining multiple datasets, and Chapter 7 describes the results of combining semantics from multiple sources.

# Chapter 6

# Sentence Planning and Discourse Structuring

## 6.1 Overview

We have shown that Neural Natural Language Generation (NNLG) models can learn to generalize different stylistic features representing a single personality using a single token representation, though more supervision leads to better results. Now we want to test if NNLG models can learn general sentence planning and discourse structuring rules, without needing to hard code these rules, as is done in Statistical Natural Language Generation (SNLG). This includes testing if these models are able to learn to control specific aggregation operations, generalize to new value combinations not seen in the training data and learn to generate discourse structures across disparate datasets.

Recent work in NNLG explicitly claims that training models end-to-end allows them to do **both** sentence planning and surface realization without the need for intermediate representations [Dusek and Jurcícek, 2016b; Lampouras and Vlachos, 2016; Mei et al., 2015; Wen et al., 2015; Nayak et al., 2017]. However, no-one has actually demonstrated that an NNLG model can faithfully produce outputs that exhibit the sentence planning and discourse operations given only examples of these operations in the training data. Given only examples in the training data, the model is not presented a structure designed to represent these features, such as tree-structured representation. Examples of these operations are in Tables 6.1, 6.2 and 6.3. Instead, NNLG evaluations focus on measuring the semantic correctness of the outputs and their fluency [Novikova et al., 2017a; Nayak et al., 2017]. The experiments in this chapter systematically test whether these claims are actually true [Reed et al., 2018].

The experiments aim to show that NNLG can perform the same operations executed by the sentence planner of a SNLG. In contrast, earlier models of SNLG for dialogue were based around the NLG architecture in Figure 6.1 [Rambow et al., 2001; Stent, 2002].

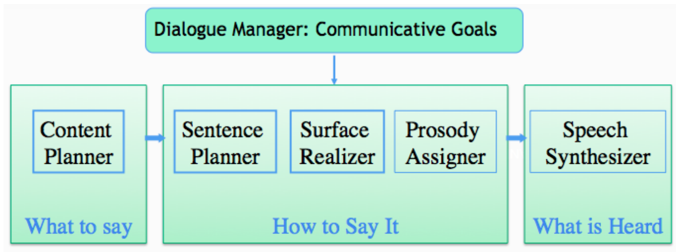


Figure 6.1: Statistical NLG Dialogue Architecture.

In statistical NLGs the dialogue manager sends one or more dialogue acts and their arguments to the NLG engine, which then makes decisions how to render the utterance using separate modules for content planning and structuring, sentence planning and surface realization [Reiter and Dale, 2000]. The sentence planner's job includes:

- **Sentence Scoping**: deciding how to allocate the content to be expressed across different sentences;
- **Aggregation**: implementing strategies for removing redundancy and constructing compact sentences;
- **Discourse Structuring**: deciding how to express discourse relations that hold between content items, such as causality, contrast, or justification.

Table 6.1 shows sample training data for an NNLG with a meaning representation (MR) for a restaurant named ZIZZI, along with three reference realizations, that should allow the NNLG to learn to realize the MR as either 1, 3, or 5 sentences. Sentence scoping affects the complexity of the sentences that compose an utterance, allowing the NLG to produce simpler sentences when desired that might then be easier for particular users to understand. Aggregation reduces redundancy, composing multiple content items into single sentences. Table 6.2 presents common aggregation operations [Cahill et al., 2001; Shaw, 1998]. Discourse structuring is often critical in persuasive settings such as recommending restaurants, hotels or travel options [Scott and de Souza, 1990; Moore and Paris, 1993], in order to express discourse relations that hold between content items. Table 6.3 shows how RECOMMEND dialogue acts can be included in the

MR, and how content can be related with JUSTIFY and CONTRAST discourse relations [Stent et al., 2002].

| # | Type | Example |
|---|---|---|
| NAME[Zizzi], PRICERANGE[MODERATE], AREA[RIVERSIDE], FOOD[English], EATTYPE[PUB], NEAR[Avalon], FAMILYFRIENDLY[NO] | | |
| 1 | 1 Sent | Zizzi is moderately priced in riverside, also it isn't family friendly, also it's a pub, and it is an English place near Avalon. |
| 2 | 3 Sents | Moderately priced Zizzi isn't kid friendly, it's in riverside and it is near Avalon. It is a pub. It is an English place. |
| 3 | 5 Sents | Zizzi is moderately priced near Avalon. It is a pub. It's in riverside. It isn't family friendly. It is an English place. |

Table 6.1: Sentence Scoping: a sentence planning operation that decides what content to place in each sentence of an utterance.

In this chapter, we systematically perform a set of controlled experiments to test whether an NNLG can learn to perform sentence planning operations. This chapter is broken up into three experiments. The first controls sentence scoping, as seen in Table 6.1, where we experiment with controlling the number of sentences in the generated output. These experiments are a proxy for controlling utterance syntactic complexity. In Table 6.1 one can see how realizing the same MR with fewer sentences (row 3) creates a simpler utterance than realizing it in 1 sentence (row 1). The second controls distributive aggregation, an aggregation operation that can compactly express a description when two attributes share the same value. This is shown in row number 6 in Table 6.2. Distributive aggregation requires learning a proxy for the semantic property of **equality** along with the standard mathematical **distributive** operation. The final experiment explores discourse contrast, which can be seen in row number 8 in

| # | Type | Example |
|---|---|---|
| NAME[THE MILL], EATTYPE[COFFEE SHOP], FOOD[ITALIAN], PRICERANGE[LOW], CUSTOMERRATING[HIGH], NEAR[THE SORRENTO] | | |
| 4 | With, Also | The Mill is a coffee shop with a *high rating* with a *low cost*, also The Mill is an Italian place near The Sorrento. |
| 5 | With, And | The Mill is a coffee shop with a *high rating* with a *high cost* and it is an Italian restaurant near The Sorrento. |
| 6 | Distributive | The Mill is a coffee shop with a **high rating and cost**, also it is an Italian restaurant near The Sorrento. |

Table 6.2: Aggregation Operation Examples.

| # | Discourse Rel'n | Example |
|---|---|---|
| NAME[BABBO], RECOMMEND[YES],FOOD[ITALIAN], PRICE[CHEAP], QUAL[EXCELLENT], NEAR[THE SORRENTO],LOCATION[WEST VILLAGE], SERVICE[POOR] | | |
| 7 | JUSTIFY ([RECOMMEND] [FOOD, PRICE, QUAL]) | I would suggest Babbo **because it serves Italian food with excellent quality and it is inexpensive**. The service is poor and it is near the Sorrento in the West Village. |
| 8 | CONTRAST [PRICE, SERVICE] | I would suggest Babbo because it serves Italian food with excellent quality and **it is inexpensive. However the service is poor**. It is near the Sorrento in the West Village. |

Table 6.3: Justify & Contrast Discourse Relations.

Table 6.3. Discourse contrast requires learning a proxy for semantic comparison. That is, that some attribute values are evaluated as positive (*inexpensive*) while others are evaluated negatively (*poor service*), and that a successful contrast can only be produced when two attributes are on opposite poles (in either order).[1]

[1]We also note that the evaluation of an attribute may come from the attribute itself, e.g. "kid friendly", or from its adjective, e.g. "excellent service".

## 6.2 Experimental Framework

### 6.2.1 Model Architecture

The goal of these experiments are to test how well NNLG models can produce realizations of these three sentence planning operations and discourse structures with varying levels of supervision, while simultaneously achieving high semantic fidelity. Figure 6.2 shows the general architecture, implemented in Tensorflow, based on TGen, an open-source sequence-to-sequence (seq2seq) neural generation framework [Abadi et al., 2016; Dusek and Jurcícek, 2016a].[2] The model uses seq2seq generation with attention [Bahdanau et al., 2014; Sutskever et al., 2014], with a sequence of LSTMs [Hochreiter and Schmidhuber, 1997] for encoding and decoding, along with beam-search and an n-best reranker.

The input to the seq2seq model is a sequence of tokens $x_t, t \in \{0, \dots, n\}$ that represent the dialogue act and associated arguments. Each $x_i$ is associated with an embedding vector $w_i$ of some fixed length. Thus, for each MR, TGen takes as input the dialogue acts representing system actions (*recommend* and *inform* acts) and the attributes and their values (for example, an attribute might be *price range*, and its value might be *moderate*), as shown in Table 6.1. The MRs (and resultant embeddings) are sorted internally by dialogue act tag and attribute name. For every MR in the training data, we have a matching reference text, which we delexicalize in pre-processing, and then re-lexicalize in the generated outputs. The encoder reads all the input vectors and

[2]https://github.com/UFAL-DSG/tgen

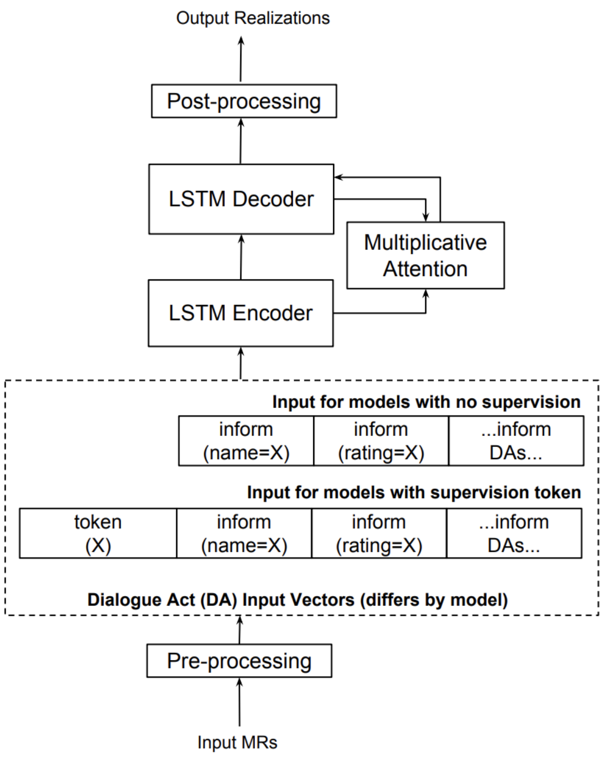


Figure 6.2: Neural Network Model Architecture, illustrating both the NO SUPERVISION baseline and models that add the TOKEN supervision.

encodes the sequence into a vector $h_n$. At each time step $t$, the encoder computes the hidden layer $h_t$ from the input $w_t$ and hidden vector at the previous time step $h_{t-1}$, following:

$$h_t = (W_1.x_t + W_2.h_{t-1}) + b$$

All experiments use a standard LSTM decoder. At each point in time, a softmax over the output of the LSTM determines the index of the next token generated. The decoder LSTM generates each word of the sentence as conditioned on the context vector $c_t$. The probability distribution $p_t$ for each candidate word at time $t$ is calculated using the following equations:

$$s_t = f(y_{t-1}, s_{t-1}, c_t)$$

and

$$p_t = softmax(s_t, y_{t-1})$$

where $s_t$ is the hidden state of the decoder LSTM at time $t$. The context vector is a weighted average of the hidden states of the encoder:

$$c_t = \sum_{k=1}^{n} \alpha_{t,k} h_k$$

$$\alpha_{t,k} = exp(e_{t,k}) / \sum_{j=1}^{n} exp(e_{t,j})$$

$$e_{t,j} = \nu(s_{t-1}, h_t)$$

where $\nu$ is usually implemented as a feed-forward network.

We test three different dialogue act and input vector representations in the three experiments, based on the level of supervision, as demonstrated by the two input vectors in Figure 6.2. **(1) models with no supervision**, where the input vector simply consists of a set of *inform* and *recommend* tokens each specifying an attribute and value pair. **(2) models with a supervision token**, where the input vector is supplemented with a new token (either *period* or *distribute* or *contrast*), to represent a latent variable to guide the NNLG to produce the correct type of sentence planning operation. **(3)**

**models with semantic supervision**, which is tested only on distributive aggregation, where the input vector is supplemented with specific instructions of which attribute value to distribute over, e.g. *low*, *average* or *high*, in the DISTRIBUTE token. We describe the specific model variations for each experiment in Sections 6.3, 6.4 and 6.5 respectively.

### 6.2.2 Corpus Creation

We use automatically generated data by PERSONAGE [Mairesse and Walker, 2011], as described in Section 3.2.1, for the sentence scoping and distributive aggregation experiments to ensure that the training data contains enough examples of particular phenomena. Then, for the contrast discourse experiment, we use crowdsourced E2E data (as described in Section 3.3) combined with data automatically generated by PERSONAGE. By automatically generating data using the SNLG PERSONAGE, this allows us to systematically create training data that exhibits particular sentence planning operations, or combinations of them. This would be difficult to find in significant amounts of in crowdsourced or natural datasets. PERSONAGE is capable of generating data with high numbers of sentences as well as generating data with specific adjective combinations for the sentence planning and distributive aggregation experiments, respectively. Also, PERSONAGE is able to generate contrast, which is uncommon in existing datasets like E2E.

We populate PERSONAGE with the syntax/meaning mappings that it needs to produce output for the E2E MRs, and then automatically produce a large (204,955 utterance/MR pairs), systematically varied sentence planning corpus.[3] This process is described in more detail in Section 3.4

### 6.2.3 Evaluation Metrics

As mentioned previously, a secondary contribution of this work is the development of new automatic evaluation metrics and corresponding scripts. These evaluation metrics are used to evaluate the semantic accuracy of the output given the semantics of the input MR. The evaluation metrics also detect the presence of the syntactic features that we are generating and their accuracy. Accuracy of syntactic features can be based on the expected behaviour given the supervision tokens or correctness. For instance, a contrast can be either correct or incorrect based on the attributes and values being contrasted. Contrast should only occur between two attributes on opposite ends of the spectrum of sentiment. It is necessary to verify the correctness of the generated contrast by confirming that the attributes on either side of the contrast cue word are not the same sentiment. We note that previous work on sentence planning has always assumed that sentence planning operations improve the quality of the output [Barzilay and Lapata, 2006; Shaw, 1998], while the primary focus here is to determine whether an NNLG can be trained to perform such operations while maintaining semantic fidelity. Moreover, due to the large size of the controlled training datasets, we observe few problems with output quality and fluency. Thus we leave an evaluation of fluency and naturalness to future work, and focus here on evaluating the multiple targets of semantic accuracy and

[3]We make the sentence planning for NLG corpus available at: nlds.soe.ucsc.edu/sentence-planning-NLG.

sentence planning accuracy.

We calculate the semantic accuracy using the SER, which is defined in Section 4.4. We also define scripts for evaluating the accuracy of the sentence planner's operations. We check whether: (1) the output has the right number of sentences; (2) attributes with equal values are realized using distributive aggregation, and (3) discourse contrast is used when semantically appropriate.

## 6.3 Sentence Scoping

First, we experiment with controlling the number of sentences in the generated output, which is measured using the *period* operator. This is to test whether it is possible to control basic sentence scoping with an NNLG. As the data for this experiment is synthetically generated, we only allow periods to be used as the end of sentence punctuation and therefore only need to confirm the presence or absence of a period to determine a sentence split. For example, Table 6.1 shows the same MR semantics realized in 1, 3 or 5 sentences using the *period* operation. We experiment with two different model supervision schemas:

- **No Supervision:** no additional information in the MR (only attributes and their values)
- **Period Count Supervision:** has an additional supervision token, PERIOD, specifying the number of periods (i.e. the number of sentences) to be used in the output realization.

For the sentence planning experiment, we generated a dataset using E2E MRs with no additional pragmatic markers and a high probability of generating a period aggregation operation versus the other aggregation operations, listed in Table 6.4. All other aggregation operations had the same likelihood of being generated.

| **Operation** | **Example** |
|---|---|
| PERIOD | *X serves Y. It is in Z.* |
| "WITH" CUE | *X is in Y, with Z.* |
| CONJUNCTION | *X is Y and it is Z. & X is Y, it is Z.* |
| ALL MERGE | *X is Y, W and Z & X is Y in Z* |
| "ALSO" CUE | *X has Y, also it has Z.* |
| DISTRIB | *X has Y and Z.* |

Table 6.4: Scoping and Aggregation Operations in PERSONAGE.

We construct a training set of 64,442 output/MR pairs and a test set of 398 output/MR pairs where the reference utterances for the outputs are generated from PERSONAGE. The maximum number of sentences in the reference for each MR is N-1 where N is the number of attributes. This is because one attribute is the name of the restaurant, which we delexicalize, and, as that is the subject of the output, it cannot be in a sentence without at least one other attribute. All other attributes are descriptors of the restaurant (proposition) and, therefore, can be on their own in a sentence. Since PERSONAGE is capable of combining many propositions into a single sentence, the minimum number of sentences for the reference of any MR is one. Table 6.5 shows the number of training instances for each MR size for each period count. The right frontier of the table shows that there are low frequencies of training instances where each proposition in the MR is realized in its own sentence (Period = Number of

MR attrs -1). The lower left quadrant of the table shows that as the MRs get longer, there are lower frequencies of utterances with Period=1.

| | | **Number of Periods** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **Attributes** | **3** | 3745 | 167 | 0 | 0 | 0 | 0 | 0 |
| | **4** | 5231 | 8355 | 333 | 0 | 0 | 0 | 0 |
| | **5** | 2948 | 9510 | 7367 | 225 | 0 | 0 | 0 |
| | **6** | 821 | 5002 | 7591 | 3448 | 102 | 0 | 0 |
| | **7** | 150 | 1207 | 2983 | 2764 | 910 | 15 | 0 |
| | **8** | 11 | 115 | 396 | 575 | 388 | 82 | 1 |

Table 6.5: Distribution of Training Data.

For all the sentence scoping experiments, we start with the default TGen parameters and monitor the losses on Tensorboard on a subset of 3,000 validation instances from the 64,000 training set. The best settings use a batch size of 20, with a minimum of 5 epochs and a maximum of 20 (with early-stopping based on validation loss). We generate outputs on the test set of 398 MRs.

### 6.3.1 Sentence Scoping Results

First, we experimented with the two types of supervision mentioned above in order to see if we could control for the number of sentences in the output. Table 6.6 shows the accuracy of both models in terms of the number of the output utterances that realize the MR attributes in the specified number of sentences. This is shows in the column labeled "Period Accuracy". The correctness of the semantics of the MR is not accounted for in this metric, which is represented by the slot error rate (SER), the metric for semantic accuracy defined in Section 4.4. This metric for both experiments show that the model is very accurate in realizing the correct semantics. In the case of NoSup, we compare the number of sentences in the generated output to those in the corresponding test reference. In the case of PeriodCount, we compare the number of sentences in the generated output to the number of sentences we *explicitly* encode in the MR. The table presents that the NoSup setting fails to output the correct number of sentences in most cases (with only a 22% accuracy), but the PeriodCount setting makes only 2 mistakes (with almost perfect accuracy), demonstrating almost complete control of the number of output sentences with the single-token supervision. We also show correlation levels with the gold-standard references (all correlations significant at $p \leq 0.01$).

| **Model** | **SER** $\downarrow$ | **Period Accuracy** $\uparrow$ | **Period Correlation** $\uparrow$ |
|---|---|---|---|
| NoSup | 0.06 | 0.216 | 0.455 |
| Period Count | 0.03 | 0.995 | 0.998 |

Table 6.6: Sentence Scoping Results.

### 6.3.2 Generalization Test

While the experiment above shows that we are able to control the number of sentences in the output, we generated the test using held-out instances from personage. Then for each MR, we only tested the model with the sentence number also generated by personage for that instance. We also wanted to confirm that regardless of the number of attributes and the MR itself we can generate any number of sentences. To this end, we carry out an additional experiment to test the ability of the PeriodCount model to see if it works more generally. To do this, we randomly select a set of 31 MRs from the test set. We then create a new test set where, for each of the 31 selected MRs, we have N-1 test instances (where N is the number of attributes). Each of these N-1 test instances have a different period token value, from 1 to N-1, in order to represent every potential number of sentences this MR could be realized with. This allows us to systematically test that, for the same MR and therefore the same semantics, we are able to control the output and generate a varied number of sentences. The new test set consists of 196 MRs.

We generated new test outputs using the model trained on the dataset with token supervision, PeriodCount. We correctly generated the desired number of sentences 84% of the time on this test set (with a correlation of 0.802 with the test refs, $p \leq 0.01$). When analyzing the errors, we observe that all of the scoping errors the model makes (31 in total) are the case of period=N-1. These cases correspond to the right frontier of Table 6.5 where there were fewer training instances. Thus, while the period supervision improves the model, it still fails on cases where there were few instances in the training set. This also demonstrates that learning to generate three sentences for an MR with four attributes is different from generating three sentences for an MR with six attributes. While we had many instances with three sentences, the test MRs with four attributes failed to generate their output in three sentences.

### 6.3.3 Complexity Experiment

We performed a third sentence scoping experiment where we specified a target sentence complexity instead of a target number of sentences. This is because sentence complexity may more intuitively correspond to a notion of reading level or difficulty, where the assumption is that longer sentences, that contain more propositions, are more complex [Howcroft et al., 2017; Siddharthan et al., 2004]. Being able to control sentence complexity would potentially allow for generation of outputs based on reading level, tailoring the output to the audience. We used the same training and test data, but labeled each reference as either high, medium or low complexity. When calculating sentence complexity, the NAME attribute does not contribute to the number of propositions in the utterance since it is the subject of the review and therefore not a proposition. The number of propositions per sentence was calculated using this formula: {number of attributes in the MR - 1} / {the number of sentences}. A reference was labeled high when there were $> 2$ propositions per sentence, medium when the number of propositions per sentence was $> 1.5$ and $\leq 2$ and low when there were $\leq 1.5$ propositions per sentence. The distribution of these labels in the training and testing data are in Table 6.7, which shows that high and medium complexity are in the training and testing data in similar proportions, with low appearing slightly less in both the training and testing data.

This experiment results in 89% accuracy across the three complexity labels. 39 of the 44 errors occurred when the labeled complexity was medium, and of those, 36 had an actual complexity of 1.5, which is on the border of medium and high. This is most

| **Complexity** | **% in Train** | **% in Test** |
|---|---|---|
| HIGH | 37.5 | 33.4 |
| MEDIUM | 37.9 | 36.9 |
| LOW | 24.6 | 29.6 |

Table 6.7: Distribution of complexity labels in the training and testing data.

likely because there is often only one sentence difference between the two complexity labels and the model struggles to make this distinction with categorical labels. This also shows that the model is learning sentence complexity not based on an absolute number of sentences given an input number of attributes.

These three experiments on sentence scoping indicate that sentence scoping can be used to create references with either exactly the number of sentences requested or categories of sentence complexity.

## 6.4 Distributive Aggregation

```
DISTRIBUTIVE AGGREGATION
if $ATTR_1 := ADJ_i$
    and $ATTR_2 := ADJ_j$
    and $ADJ_i = ADJ_j$
then DISTRIB$(ATTR_1, ATTR_2)$
```

Figure 6.3: Semantic operations underlying distributive aggregation.

Aggregation describes a set of sentence planning operations that combine multiple attributes into single sentences or phrases. Common aggregation operations can be seen in Table 6.4. In this section, we focus on distributive aggregation as defined in

Figure 6.3 and illustrated in Row 6 of Table 6.2. Distributive aggregation occurs when there are two attributes (nouns) with the same values (adjectives). Instead of repeating the values, for example, "The restaurant has a high price and a high rating", both attributes can be associated with a single mention of the value, as in "The restaurant has a high price and rating". In a SNLG setting, the generator achieves this type of aggregation by operating on syntactic trees [Shaw, 1998; Scott and de Souza, 1990; Stent et al., 2004; Walker et al., 2002a]. In an NNLG setting, we hope the model will induce the syntactic structure and the mathematical operation underlying it, automatically, without explicit training supervision.

We again created a novel training corpus for these examples using PERSONAGE and we again used the E2E ontology, like in the previous sentence scoping experiment. However, we changed values to allow more opportunities for distributive aggregation, as there were low frequencies of distributive aggregation in the original data. This was because lexical values rarely matched for different attributes. To prepare the training data, we limited the values for PRICE and RATING attributes to LOW, AVERAGE, and HIGH. We reserved the combination {PRICE=HIGH, RATING=HIGH} for test, leaving two combinations of values where distributive aggregation is possible ({PRICE=LOW, RATING=LOW} and {PRICE=AVERAGE, RATING=AVERAGE}). This allowed us to determine if the model is only learning to generate combinations of words that are in the training data or if the model can learn the sentence structure of distributive aggregation. We then used all three values in MRs where the values associated with price and rating are not the same {PRICE=LOW, RATING=HIGH}. This ensured that the model *does* see the value HIGH in training, but never in a setting where distributive aggregation is possible. We always distributed whenever possible, so every MR where the values are the same used distributive aggregation. Aggregation operations between all other attributes used the other operations defined in PERSONAGE, as specified in Table 6.4, with equal probability.

The aggregation training set contains 63,690 total instances, with 19,107 instances for each of the two combinations that are able to distribute and 4,246 instances for each of the six combinations that are not able to distribute. The test set contains 408 MRs, 288 specify distributive aggregation over HIGH,[4] 30 specify distributive aggregation over AVERAGE, 30 over LOW, and 60 are examples where the values are different and therefore do not require distributive aggregation (NONE). We test whether the model will learn the equality relation independent of the value (HIGH vs. LOW), and thus realizes the aggregation with HIGH. For this experiment, we test three different types of supervision with which we train the models:

- **No Supervision:** no additional information in the MR (only attributes and their values)
- **Binary Supervision:** we add a supervision token, DISTRIBUTE, where the value is either a binary 0 or 1 indicating whether or not the corresponding reference text contains an aggregation operation over attributes *price range* and *rating*.
- **Semantic Supervision:** we add a supervision token, DISTRIBUTE, where the

[4]which again we note is *not* a setting seen in the training data, and explicitly tests the models' ability to generalize

value is either a string that is NONE if there is no aggregation over *price range* and *rating* in the corresponding reference text, or a value of LOW, AVERAGE, or HIGH for distributive aggregation.

Like in the previous experiment on sentence scoping, we started with the default TGen parameters and monitor the losses on Tensorboard on subset of 3,000 validation instances from the 63,000 training set. The best settings use a batch size of 20, with a minimum of 5 epochs and a maximum of 20 epochs with early-stopping.

### 6.4.1 Distributive Aggregation Results

| **Model** | **SER** $\downarrow$ | **Distributive Accuracy** $\uparrow$ | **Distributive Accuracy (on HIGH)** $\uparrow$ |
|---|---|---|---|
| NoSup | 0.12 | 0.29 | 0.00 |
| Binary | 0.07 | 0.99 | 0.98 |
| Semantic | 0.25 | 0.36 | 0.09 |

Table 6.8: Distributive Aggregation Results.

Table 6.8 shows the accuracy of each model overall on all 4 values, as well as the accuracy specifically on HIGH, the only distributive aggregation value unseen in the training data. The NOSUP model has a low accuracy overall, and is unable to generalize to HIGH, which is unseen in the training data. It is able to use the HIGH value frequently, but is not able to distribute (generating output like *high cost and cost*).

The BINARY model is the best performing model, with an almost perfect accuracy. It is able to accurately distribute over LOW and AVERAGE for every test instance, however it makes some mistakes when trying to distribute over HIGH; specifically, while it always distributes, it sometimes uses an incorrect value (LOW or AVERAGE). Whenever BINARY correctly distributes over HIGH it, interestingly, always selects the attribute *rating* before *price range*, realizing the output as *high rating and price.* Additionally, BINARY is consistent even when it incorrectly uses the value LOW instead of HIGH: it always selects the attribute *price range* before *rating.*

Surprisingly, the SEMANTIC model does poorly, with 36% overall accuracy. For the value HIGH, there is only a 9% accuracy, where most of the mistakes on HIGH involve repeating the attribute *high rating and rating*, including examples where it does not distribute at all, e.g. *high rating and high rating.* We believe the poor results from SEMANTIC are due to the model not seeing the value HIGH in relation to the supervision token in the training data and therefore does not encode the value HIGH, unlike for values that are seen in the training data, which it can recognize are related such as the value LOW and the word “low”. In other words, even though the model knows the word “high”, the value HIGH in relation to the semantic supervision attribute is completely unknown to the model and it does not provide any useful information to help the model realize it needs to use distributive aggregation.

We also completed a human evaluation to compare the fluency, grammaticality, and semantic coherency of the outputs of the BINARY model. We described these terms in more detail in Section 4.3. We had five Turkers select which was of two outputs from the model was more fluent, grammatical and coherent. We split the output into three categories: (1) high distributive (`high`), where the output realized the “high” adjective for both attributes and used the distributive aggregation operation, (2) low and average distributive (`lowaverage`), where the output realized either the "low" or "average" adjective for both attributes and used the distributive aggregation operation, and (3) not distributive (`none`), where the output realized different adjectives for both attributes and did not use the distributive aggregation operation. For each pair of outputs shown to the Turkers, we put together utterances from two different categories and asked them which they thought was better for each category we evaluated. We found, when compared to both `lowaverage` and `none`, `high` was preferred across all categories. But between `lowaverage` and `none`, LOWAVERAGE is preferred across all categories. This indicates that using the distributive operation creates an utterance that humans prefer in all measured aspects. Also, even though "high" is the only value not seen distributed in the training data, the output which uses this adjective is preferred by humans. This could be because "high" rating is seen as positive. This does demonstrate that the model generated good quality output even on unseen data, since it did not detract from the readers' opinions of the utterance. Results in Table 6.9.

| **Utt.Type A** | **Utt. Type B** | NAT. | | COHER. | | GRAMMAT. | |
|---|---|---|---|---|---|---|---|
| | | A | B | A | B | A | B |
| `high` | `low&average` | 57% | 43% | 67% | 33% | 57% | 43% |
| `high` | `none` | 67% | 33% | 60% | 40% | 63% | 37% |
| `low&average` | `none` | 63% | 37% | 63% | 37% | 63% | 37% |
| AGREEMENT | | 0.58 | | 0.53 | | 0.54 | |

Table 6.9: Human Evaluation on Mechanical Turk comparing `high`, `lowaverage` and `none` for Naturalness, Semantic Coherence, and Grammaticality.

## 6.5 Contrast

Persuasive settings such as recommending restaurants, hotels or travel options often have a critical discourse structure [Scott and de Souza, 1990; Moore and Paris, 1993; Nakatsu, 2008]. For example, a recommendation may consist of an explicitly evaluative utterance, e.g. *Chanpen Thai is the best option*, along with content related by the justify discourse relation, e.g. *It has great food and service*, as can be seen Table 6.3 or a recommendation may explicitly contrast two different attributes when one is perceived positive and the other negative.

The experiments in this section focus on DISCOURSE-CONTRAST, which is defined in Figure 6.4. Contrast can occur between two attributes when the value of one of the attributes is considered negative and the value of the other attribute is considered positive. We developed a script to find contrastive sentences in the 40K E2E training set by searching for any instance of a contrast cue word, such as *but*, *although*, and *even if*. This script identified 3,540 instances. While this data size is comparable to the 3-4K instances used in prior work [Wen et al., 2015; Nayak et al., 2017], we anticipated that it might not be enough data to properly test whether an NNLG can learn to produce discourse contrast. We were also interested in testing whether synthetic data would

| DISCOURSE CONTRAST |
|---|
| if EVAL($ADJ_i(ATTR_1)$) = POS |
| and EVAL($ADJ_j(ATTR_2)$) = NEG |
| then CONTRAST($ATTR_1, ATTR_2$) |

Figure 6.4: Semantic operations underlying discourse contrast.

improve the ability of the NNLG to produce contrastive utterances while maintaining semantic fidelity. Thus, we used PERSONAGE to generate the NYC data described in Section 3.2 in order to generate an additional 3,500 examples of one form of contrast using only the discourse marker *but*. This was the most common contrast cue in the E2E data. This also allows us to test if, given a dataset with few instances of contrast, we can improve the model's ability to generate contrast on MRs from this dataset by augmenting the training data with contrastive data from another dataset.

Table 6.10 illustrates both PERSONAGE and E2E contrast examples. While PERSONAGE also contains JUSTIFICATIONS, which could possibly confuse the NNLG, it offers many more attributes that can be contrasted and, thus, more unique instances of contrast. We also delexicalize the attributes cuisine and location in both datasets. We do this as the values for neither attribute have explicit sentiment and they were both simple to delexicalize, which gives the model fewer values to learn and improves its chance to learn contrast. We create four training datasets with contrast data in order to systematically test the effect of combining training data from different sources. Table 6.11 provides an overview of the training sets.

**3K Training Set.** This dataset consists only of instances of contrast in the E2E training data (i.e. 3,540 E2E references and no additional data).

**7K Training Set.** We created a training set of 7K references by supplementing the E2E contrastive references with an equal number of PERSONAGE references with contrast.

**11K Training Set.** Since 7K is a smaller dataset size than is desirable for training an NNLG, we created several additional training sets with the aim of helping the model

| Source | MR | Realization |
|---|---|---|
| NYC | name[xname] recommend[no] cuisine[xcuisine] decor[bad] qual[acceptable] location[xlocation] price[affordable] service[bad] | I imagine xname isn't great because xname is affordable, but it provides bad ambiance and rude service. It is in xlocation. It's a xcuisine restaurant with acceptable food. |
| E2E | name[xname] cuisine[xcuisine] location[xlocation] familyFriendly[no] | It might be okay for lunch, but it's not a place for a family outing. |
| E2E | name[xname] eatType[coffee_shop] cuisine[xcuisine] price[more_than_$30] customerRating[low] location[xlocation] familyFriendly[yes] | Xname is a low customer rated coffee shop offering xcuisine food in the xlocation. Yes, it is child friendly, but the price range is more than $30. |

Table 6.10: Training examples of E2E and NYC Contrast sentences.

| Training Sets | NYC #N | E2E #N |
|---|---|---|
| 3K | N/A | 3,540 contrast |
| 7K | 3,500 contrast | 3,540 contrast |
| 11K | 3,500 contrast | 3,540 contrast + 4K random |
| 21K | 3,500 contrast | 3,540 contrast + 14K random |
| 21K CONTRAST | 3,500 contrast | 3,540 contrast + 14K random |

Table 6.11: Overview of the training sets for contrast experiments.

learn to correctly realize domain semantics while still being able to produce contrastive utterances. Thus, we added an additional 4K crowd-sourced E2E data that was not contrastive to the training data, for a total of 11,065. See Table 6.11.

**21K Training Set.** Additionally, we created a larger training set by adding more E2E data, again to test the effect of increasing the size of the training set on realization of domain semantics, without a significant decrease in the ability to produce contrastive utterances. We added an additional 14K E2E references, for a total of 21,065. See

Table 6.11. We perform two experiments with the 21K training set. First, we trained on the MR and reference exactly as we had done for the 7K and 11K training sets. The second experiment added a contrast token during training time with values of either 1 (contrast) or 0 (no contrast) to test if added supervision would help the model achieve better control of contrast.

There is also a representation of the distribution of data from NYC and E2E (contrast data and non-contrast data) in each training dataset in Figure 6.5.

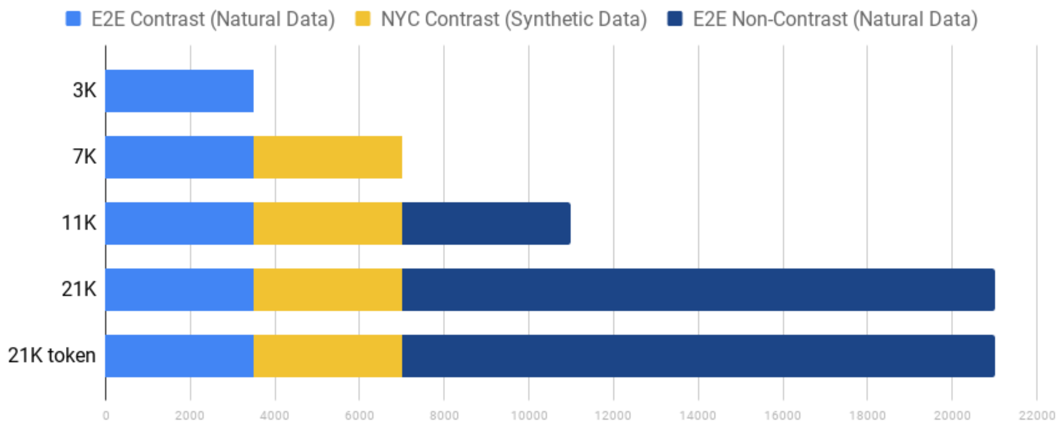


Figure 6.5: The Distribution of the Contrast Training Datasets.

**Contrast Test Sets.** To have a potential for contrast, there must be an attribute with a positive value and an additional attribute with a negative value in the same MR. We constructed three different test sets, two for E2E and one for NYC. First, we created a delexicalized version of the test set used in the E2E generation challenge. We delexicalized the test sets so they matched the already delexicalized training dataset. This resulted in a test of 82 MRs of which only 25 could support contrast (E2E Test). In order for an MR to support contrast it must have two attributes from Table 6.12, one that has a positive value and one that has a negative value. It can then have any other additional attribute and value pairs, including other attributes and values which support contrast. In order to allow for a better test of contrast, we constructed an additional test set of 500 E2E MRs which all could support contrast (E2E CONTRAST TEST). For the NYC test, which provides many opportunities for contrast, we created a dataset of 785 MRs that were different than those seen in the training data (NYC TEST), 374 of which support explicit contrast. We utilize the contrast token, at test time, for the 21K contrast token experiment, to match the training data.

While these experiments focus on contrast between two attributes that have clear sentiment, such as customer rating, it is generally agreed that a higher rating is better, sentiment towards attributes often depends on the opinions of the author. Attributes other than the ones listed in Table 6.12 can potentially support contrast if the author has a negative or positive opinion of them. Location, for example, can be very important to some people. An utterance like “has good food but is located in riverside” can be a very reasonable utterance from someone who dislikes riverside. Also, we made the assumption that *price range* being high was negative, but some people prefer higher end restaurants and price might be indicative of this. For these experiments, we do not consider the author’s opinion, and therefore do not count these as correct contrast. This is so we can measure the model’s ability to correctly generate contrast based on what it has seen and not generate contrast that it not prevalent in the data.

| **Attribute** | **Positive Values** | **Negative Values** | **Dataset** |
|---|---|---|---|
| price | cheap, low, less than $20 | very expensive, expensive, high, more than $30 | both |
| customerRating | high, 5 out of 5 | low, 1 out of 5 | E2E |
| familyFriendly | yes | no | E2E |
| service | fantastic, good | bad, terrible | NYC |
| qual | fantastic, good | bad, terrible | NYC |
| decor | fantastic, good | bad, terrible | NYC |

Table 6.12: Attributes with sentiment and their positive and negative values.

### 6.5.1 Contrast Results

We present the results for both SER and contrast for the Original E2E test set in Table 6.13, E2E Contrast in Table 6.14, and NYC test set in Table 6.15.

Table 6.13 shows the results for testing on the Original E2E test set, where we only have 25 instances out of 82 with the possibility for contrast. Overall, the table shows large performance improvements with the CONTRAST token supervision for the model trained on 21K instances for both SER and correct contrast. On the Original E2E test set, the model trained on just 3K E2E instances gives a slot error rate (SER) of 0.38 and only 15% of the attempts at contrast were correct. We define a correct contrast as a contrast between a negative and positive attribute-value pair. The model trained on 7K instances, which was supplemented with additional synthetically generated contrast examples, gets a correct contrast of 0.41 but the SER is much worse, increasing from 0.38 to 0.56. Interestingly, the model trained on 11K instances performed better than the model trained on 3K instances for contrast correct. This occurs even though the model trained on 11K has a smaller proportion of contrast instances, suggesting a positive effect for the automatically generated contrast examples along with more E2E training data. The model trained on 21K instances without additional supervision has the least number of attempts to contrast since the frequency of contrast data is low, but with the CONTRAST token, it attempts contrast every time it is possible (25/25 instances) and has the highest percentage of correct contrast out of all the models by 34%.

| **Train** | **E2E Test (N = 82)** | | |
|---|---|---|---|
| | SER ↓ | CONTRAST ATTEMPTS ↑ | CONTRAST CORRECT ↑ |
| 3K | .38 | 13 | .15 |
| 7K | .56 | 61 | .41 |
| 11K | .31 | 24 | .33 |
| 21K | .28 | 2 | .50 |
| 21K CONTRAST | **.24** | 25 | **.84** |

Table 6.13: Slot Error Rates and Contrast for Original E2E.

In Table 6.14, we can see the results on the test set E2E Contrast, where every test instances has the potential for contrast. For this test set, we also see similar trends as when we tested on Original E2E, with the lowest SER (0.16) and highest correct contrast (0.75) ratios for the experiment with token supervision on 21K instances. Again, we see a much better performance from the model trained on 11K instances than the model trained on 3K instances or the model trained on 7K instances. This is in terms of both SER and correct contrast, indicating that more training data (even if that data does not contain contrast) helps the model. As before, we see a very low amount of contrast attempts with the model trained on 21K instances without additional supervision, with a huge increase in the number of contrast attempts when using token supervision

(422/500).

| **Train** | **E2E Contrast Test (N=500)** | | |
|---|---|---|---|
| | SER ↓ | CONTRAST ATTEMPTS ↑ | CONTRAST CORRECT ↑ |
| 3K | .70 | 213 | .19 |
| 7K | .45 | 325 | .22 |
| 11K | .23 | 227 | .70 |
| 21K | .17 | 13 | .62 |
| 21K CONTRAST | **.16** | 422 | **.75** |

Table 6.14: Slot Error Rates and Contrast for E2E, Contrast Only.

Table 6.15 also shows large performance improvements with CONTRAST token supervision for the NYC test set, again with improvements for the 21K CONTRAST in both SER and in correct contrast. Interestingly, while we get the highest correct contrast ratio of 0.85 with 21K CONTRAST, we actually see *fewer* contrast attempts. This shows that the most explicitly supervised model becomes more selective when deciding when to generate contrast. With the model trained on 7K instances, the neural model **always** produces a contrastive utterance for the NYC MRs (all the NYC training data is contrastive), although 35% of the time this contrast is incorrect. Although the model trained on 21K instances with CONTRAST supervision is not trained with any NYC non-contrastive MRs, the additional E2E training data allows it to improve its ability to decide when to contrast as well as improving the SER in the final experiment. This is especially interesting since for each addition of E2E non-contrasting data, both the SER and the contrast accuracy improve for NYC. This shows that more data improves an NNLG regardless of its direct correlation to test.

| **Train** | **NYC Test (N = 785)** | | |
|---|---|---|---|
| | SER ↓ | CONTRAST ATTEMPTS ↑ | CONTRAST CORRECT ↑ |
| 3K | N/A | N/A | N/A |
| 7K | .29 | 784 | .65 |
| 11K | .26 | 696 | .71 |
| 21K | .25 | 659 | .82 |
| 21K CONTRAST | .24 | 566 | **.85** |

Table 6.15: Slot Error Rates and Contrast for NYC.

We also performed human evaluation on the outputs from the 21K CONTRAST model using the E2E contrast test set. While every MR had the potential for contrast and the supervision indicated there should be contrast in every utterance, 78 outputs did not have contrast. We compared these 78 instances where contrast failed to a random sample of 78 outputs with correct contrast to see how the contrast learned through transfer learning affected the readers' experience. We compared the utterances in terms of fluency, grammaticality and semantic coherency, as described in Section 4.3. We had five Turkers evaluate each utterance pair. We found that, across all three metrics, that the outputs without contrast were evaluated as better. This shows that there are still improvements to be made to improve the quality of the contrast being generated. Results are in Table 6.16

## 6.6 Summary

These experiments demonstrate that we can learn a general sentence planning framework from specific examples in the training data with minimal supervision and

| | Nat. | Coher. | Grammat. |
|---|---|---|---|
| contrast | 35.9% | 32.1% | 23.1% |
| no contrast | 64.1% | 67.9% | 76.9% |
| agreement | 0.56 | 0.56 | 0.58 |

Table 6.16: Human Evaluation on Mechanical Turk comparing outputs with contrast and without contrast for Naturalness, Semantic Coherence, and Grammaticality (N=78). Results are percentage of times Turkers rated the contrast or without contrast as "better".

thereby control the sentence planning operations expressed in output, learn how to apply sentence planning operations to new values that the model was not trained on, and blend data from multiple sources to improve the ability to learn sentence planning operations which might be underrepresented in existing datasets. In the sentence scoping experiment, we had the model learn how to split the same MR into different numbers of sentences, and apply that to MRs that it had never seen before. For the distributive experiment, we trained the model to distribute adjectives and, by using a binary token, the model learned to distribute on a value that it had seen, but never seen distributed in the training data. This showed us that in order to generalize, additional supervision is sometimes necessary. In the contrast experiment we combined data from two sources to augment the number of times the model sees contrast. This helped both sources learn to contrast more accurately and improved semantic accuracy for both sources, even when we only added data from one source.

Overall, the results show that the models benefit from extra latent supervision, which improves the **semantic accuracy** of the NNLG, provides the capability to **control** variation in the output, and enables **generalizing** to unseen value combinations.

These are forms of generalization that we believe can extend to other types of sentence planning operations and other forms of stylistic variation.

# Chapter 7

# Source Blending

## 7.1 Overview

Current Natural Language Generators (NLGs) take massive amounts of data to train. The creation of such datasets is labor-intensive and time-consuming. We believe when building an NLG for a new domain ontology, it should be possible to re-use data built on existing domain ontologies. If this idea is possible, it would speed up the development of new dialogue systems significantly.

In this chapter, we built a new domain ontology based on **combining** two different existing ontologies within the same domain and utilizing their training data. Each dataset is based on a different domain ontology in the restaurant domain. They each have novel attributes and dialogue acts not seen in the other dataset, e.g., only one dataset had attributes representing *family-friendly* and *rating* information and only the other dataset had attributes for *decor* and *service*. The aim was an NLG engine that can

| ID | Ontology | MEANING REPRESENTATION | EXAMPLE |
|---|---|---|---|
| E1 | NYC (TRAIN) | RECOMMEND[YES], INFORM(NAME[RESTAURANT], SERVICE[EXCELLENT], FOOD[EXCELLENT], DÉCOR[EXCELLENT], LOCATION[AREA], PRICE[EXPENSIVE]) | I suggest you go to [RESTAURANT]. The food, service and atmosphere are all excellent, even if it is expensive. Its in [AREA]. |
| E2 | E2E (TRAIN) | INFORM(NAME[RESTAURANT], EATTYPE[RESTAURANT-TYPE], CUSTOMER-RATING[HIGH], AREA[AREA], NEAR[POINT-OF-INTEREST]) | [RESTAURANT] is a [RESTAURANT-TYPE] in [AREA] near [POINT-OF-INTEREST]. It has a high customer rating. |
| E3 | COMBINED (TEST) | RECOMMEND = YES, INFORM(NAME[RESTAURANT], EATTYPE[RESTAURANT-TYPE], FOOD = EXCELLENT, LOCATION[AREA], NEAR[POINT-OF-INTEREST], CUSTOMER-RATING[HIGH], DÉCOR = EXCELLENT, SERVICE=EXCELLENT, PRICE=EXPENSIVE) | [RESTAURANT] is the best because it has excellent service and atmosphere. It is a [RESTAURANT-TYPE] offering excellent food in [AREA] near [POINT-OF-INTEREST] with a high customer rating, but it is expensive. |

Figure 7.1: E1 and E2 illustrate training instances from the two source datasets E2E and NYC. E2E attributes are represented in blue and NYC is in red. Some attributes are shared between both sources: here the unique dialogue acts and attributes for each source are underlined in E1 and E2. E3 illustrates an MR from the target test set that we dub COM. All the MRs in COM combine dialogue acts and attributes from E2E and NYC. There is no training data corresponding to E3. The MRs illustrate how some attribute values, e.g. RESTAURANT NAME, POINT-OF-INTEREST, are delexicalized to improve generalization.

realize utterances for the extended **combined** ontology not found in the training data, e.g. for meaning representations (MRs) that specify values for *family-friendly*, *rating*, *decor* and *service*. Figure 7.1 illustrates this task. Example E1 is from a dataset referred to as NYC, described in Section 3.2, while E2 is from the dataset created for the E2E NLG shared task, described in Section 3.3. As we explain in detail in Section 7.2, E1 and E2 are based on two distinct ontologies. Example E3 illustrates the task addressed in this paper: we created a test set of novel MRs for the combined ontology and trained a model to generate high-quality outputs where individual sentences realize attributes from both ontologies.

To our knowledge, this is a entirely novel task. While it is common practice in NLG to construct test sets of MRs that realize attribute combinations not seen in the training data, initial experiments showed that this task is surprisingly adversarial. However, methods for supporting this type of generalization and extension to new cases would greatly benefit task-oriented dialogue systems. In these NLG systems, it is common to start with a restricted set of attributes and then slowly enlarge the domain ontology over time. Also, in real world data, new attributes are constantly being added to databases of restaurants, hotels, and other entities to support better recommendations and better search. The experiments tested whether an NLG for an enlarged ontology can be produced using individual existing datasets that each only cover a subset of attributes.

We describe below how we created a test set — that we call COM — of combined MRs to test different methods for creating such an NLG. A baseline sequence-to-sequence NLG model had a slot error rate (SER) (Section 4.4) of 0.45 and only produced semantically perfect outputs 3.5% of the time. We experimented with three different ways of conditioning the model by incorporating *side constraints* that encode the source of the attributes in the MR [Sennrich et al., 2016; Harrison et al., 2019] to improve performance. However, this only increases the proportion of semantically perfect model outputs from 3.5% to 5.5% (Section 7.4.1).

We then proposed and motivated a novel self-training method that dramatically improves performance by learning from the model mistakes. Error analysis showed that the models **do** produce many **combined** outputs but with errorful semantics. We developed a rule-based text-to-meaning semantic extractor, based on SER, that automatically creates novel correct MR/text training instances from errorful model outputs. Self-training experiments used these corrected MR/text instances, thus learning from mistakes (Section 7.4.2). We validated the text-to-meaning extractor with human evaluation. We found that a model trained with this process produced SERs of only 0.03 and semantically perfect outputs 81% of the time (a 75.4 percent improvement). We also found that using a pre-trained model, GPT-2 [Radford and Narasimhan, 2018], does not improve the semantic results but has more stylistic variation in the final model output (Section 7.5.3). A human evaluation showed that these outputs are also natural, coherent, and grammatical. The contributions are:

- Definition of a novel generalization task for neural NLG engines, which generates from unseen MRs that combine attributes from two datasets with different ontologies;
- Systematic experiments on methods for conditioning NLG models, with results showing the effects on model performance for both semantic errors and combining

attributes;

- A novel self-training method that learns from the model's mistakes to produce semantically correct outputs 81% of the time, an absolute 75.4% improvement.

We start in Section 7.2 by defining the task in more detail, we describe the models and metrics in Section 7.3, and we provide results in Section 7.4. In Section 7.5, we test if using a pre-trained model, GPT-2, improves the results [Reed et al., 2020].

## 7.2 Ontology Merging and Data Curation

We started with two existing datasets, NYC and E2E, representing different ontologies for the restaurant domain. We describe Both of these datasets in more detail in Chapter 3. The NYC dataset used the `narrator` voice with minimal pragmatic markers and an even distribution of aggregation operations. We used the crowdsourced E2E dataset, combining training and development datasets into a single, larger, training dataset. Figure 7.1 shows sample MRs for each source and corresponding training instances as E1 and E2.

### 7.2.1 Ontology Merging

We first made a new combined ontology, ONTO-COM, by merging NYC and E2E. Attributes, dialogue acts, and sample values for E2E and NYC are illustrated on the left-hand side of Figure 7.2. The result of merging them to create the new ontology is on the right-hand side of Figure 7.2. Since there are only eight attributes in each source dataset, we developed a script by hand that maps the MRs from each source into the ONTO-COM ontology.

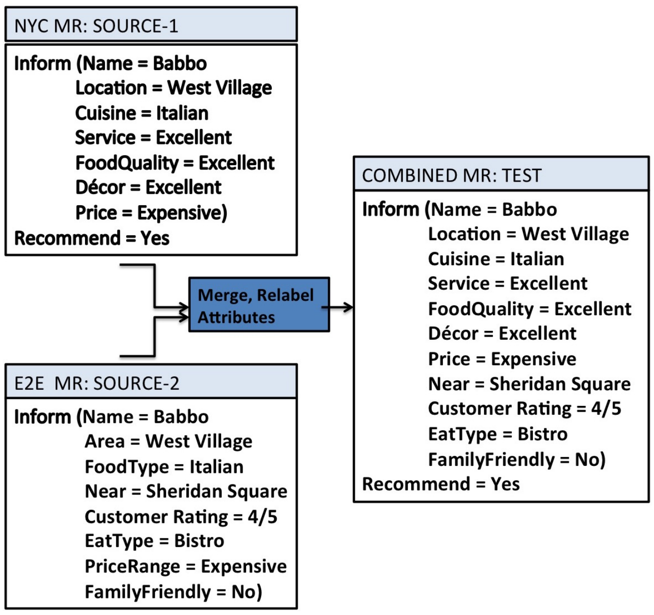


Figure 7.2: An example illustrating how dialogue acts and attributes for both source databases were merged and relabelled to make a new combined ontology used in the training and testing data.

As Figure 7.2 shows, both datasets have the INFORM dialogue act and include the attributes *name*, *cuisine*, *location*, and *price* after mapping. The unique attributes for the NYC ontology are scalar ratings for *service*, *food quality*, and *decor*. The NYC dataset also has the RECOMMEND dialogue act, seen in E1 in Figure 7.1. The unique attributes of the E2E ontology are *customer rating*, *eat type* ("coffee shop"), *near* and *family-friendly*.

### 7.2.2 Dataset Curation

**Training Dataset.** Given the combined ontology ONTO-COM, we then mapped the training data for both E2E and NYC into ONTO-COM by relabelling the MRs to have consistent names for shared attributes, as illustrated in Figure 7.2. We created a balanced training set of ∼77K MR/utterance pairs from the two original datasets by combining all NYC references with a random same-size sample of E2E references.

**Test Set.** We then manually created a test set, COM, consisting of 3040 MRs based on the new combined ontology ONTO-COM. Each test MR must have at least one attribute unique to E2E and one attribute unique to NYC. This is so the MR combines attributes from both sources, and these MRs form a dataset that is composed of combinations never seen in training.[1] Example E3 in Figure 7.1 provides an example test MR. The procedure for creating the test set ensured that its length and complexity are systematically varied, with lengths normally distributed and ranging from three to ten attributes. Recommendations only occur in the NYC training data, and they increase both **semantic** and **syntactic** complexity, with longer utterances that use the discourse relation of JUSTIFICATION [Stent et al., 2002], e.g., *Babbo is the best* **because** *it has excellent food.* We hypothesize that recommendations may be more challenging to combine across domains, so we vary MR complexity by including the RECOMMEND dialogue act in half the test references. We show in Section 7.4.2 that the length and complexity of the MRs are important factors in the performance of the trained models.

[1]The train and test data are available at http://nlds.soe.ucsc.edu/source-blending-NLG

## 7.3 Experimental Overview and Methods

Given the training and test sets for the combined ontology in Section 7.2, we tested four different neural model architectures and present results in Section 7.4.1. We then proposed a novel self-training method and presented results in Section 7.4.2. These experiments relied on the model architectures presented here in Section 7.3.1 and the Text-to-Meaning semantic extractor and performance metrics in Section 7.3.2.

### 7.3.1 Model Architectures

Here we also used a standard RNN encoder–decoder model [Sutskever et al., 2014] that mapped a source sequence (the input MR) to a target sequence (the utterance text). We first implemented a baseline model and then added three variations of model supervision that aim to improve semantic accuracy. All of the models were built with OpenNMT-py, a seq2seq modeling framework [Klein et al., 2017].

#### 7.3.1.1 Encoder

The MR was represented as a sequence of (attribute, value) pairs with separate vocabularies for attributes and values. Each attribute and each value were represented using 1-hot vectors. An (attribute, value) pair was represented by concatenating the two 1-hot vectors.

The input sequence was processed using two single layer bidirectional-LSTM [Hochreiter and Schmidhuber, 1997] encoders. The first encoder operated at the pair level, producing a hidden state for each attribute-value pair of the input sequence. The second LSTM encoder was intended to produce utterance level context information in the form of a full MR encoding produced by taking the final hidden state after processing the full input sequence. The outputs of both encoders were combined via concatenation. That is, the final state of the second encoder was concatenated onto each hidden state output by the first encoder. The size of the pair level encoder was 46 units and the size of the MR encoder was 20 units. Model parameters were initialized using Glorot initialization [Glorot and Bengio, 2010] and optimized using Stochastic Gradient Descent with mini-batches of size 128.

#### 7.3.1.2 Decoder

The decoder was a uni-directional LSTM that uses global attention with input-feeding. Attention weights were calculated via the *general* scoring method [Luong et al., 2015]. The decoder took two inputs at each time step: the word embedding of the previous time step, and the attention weighted average of the encoder hidden states. The ground-truth previous word was used when training, and the predicted previous word when evaluating. Beam search with five beams was used during inference.

#### 7.3.1.3 Supervision

Figure 7.3 shows the baseline system architecture as well as three types of supervision, based on conditioning on source (E2E, NYC) information. The additional supervision was intended to help the model attend to the source dataset information. We call the three types of supervision GUIDE, ATTR and BOOL, and the baseline architecture

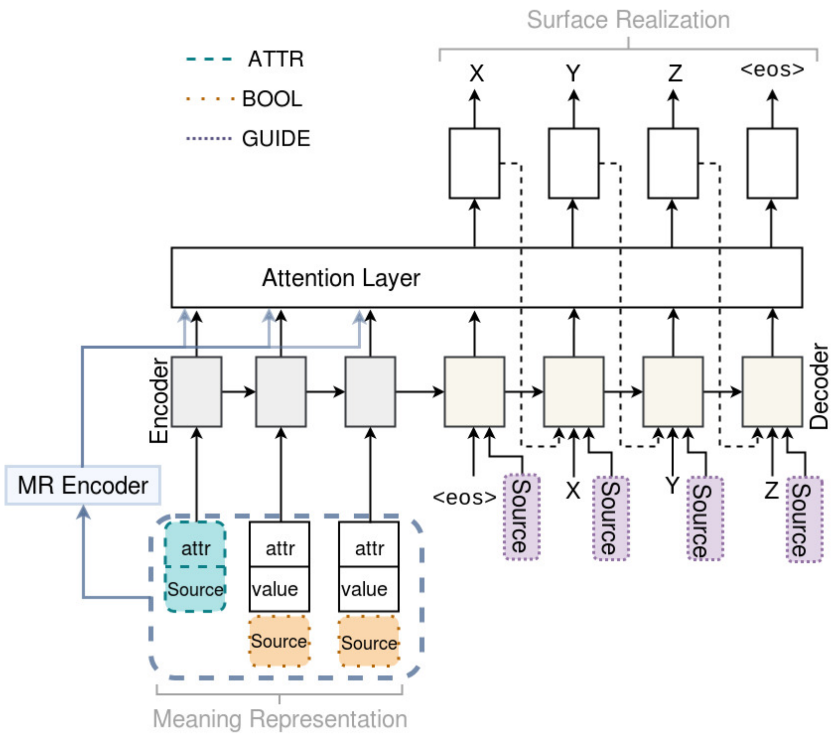


Figure 7.3: Attentional Encoder-Decoder architecture with each supervision method shown.

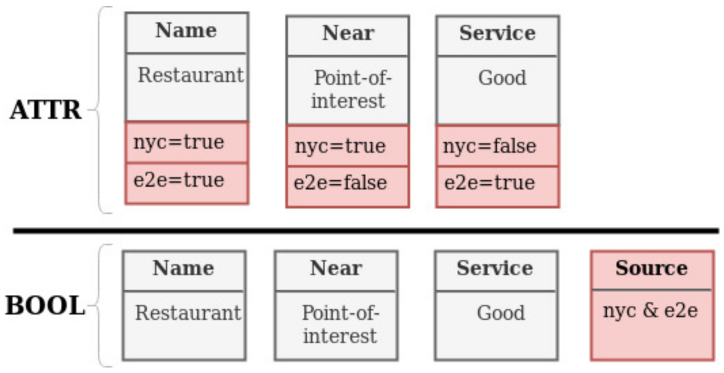


Figure 7.4: An illustration of ATTR and BOOL supervision methods, with the source supervision (NYC or E2E) shown in red.

NOSUP, representing that it has no additional supervision.

The supervision methods are shown in Figure 7.4. The source feature had a vocabulary of three items: *nyc*, *e2e* and *both*. Since *both* was never seen in the training data, the source information was represented using two booleans: *True*||*False* denoted a reference from E2E while *False*||*True* denoted a reference from NYC. This encoding was intended to encourage generalization at inference time. During inference, blending of information from both sources was specified by using *True*||*True*. The ATTR supervision method represented the source information by concatenating the boolean source token onto each attribute as seen in the top half of Figure 7.4. This redundantly represented the source information locally to each attribute, which has been effective for tasks such as question generation and stylistic control [Harrison and Walker, 2018; Harrison et al., 2019]. The BOOL supervision method added the boolean source token to the end of the sequence of attribute-value pairs as its own attribute, as in work on machine translation and controllable stylistic generation [Sennrich et al., 2016; Yamagishi et al., 2016; Ficler and Goldberg, 2017]. This can also be seen in the bottom half of Figure 7.4. The GUIDE supervision method inputted the source information directly to the decoder LSTM. In previous work, putting information into the decoder in this way yielded improvements in paraphrase generation and controllable generation [Iyyer et al., 2018; Harrison et al., 2019].

### 7.3.2 Text-to-Meaning Semantic Extractor

Much previous work in NLG relied on a test set that provides gold reference outputs, and then applied automatic metrics such as BLEU that compare the gold reference to the model output [Papineni et al., 2002; Dušek et al., 2020]. This occurred even though the limitations of BLEU for NLG are widely acknowledged. We addressed this in more detail in Chapter 4. To address these limitations, recent work has started to develop "referenceless" NLG evaluation metrics [Dusek et al., 2017; Kann et al., 2018; Tian et al., 2018; Mehri and Eskenazi, 2020].

Since there are no reference outputs for the COM test set, we needed a referenceless evaluation metric. We developed a rule-based text-to-MR semantic extractor (TTM) that allowed us to automatically construct an MR from an NLG model's textual output by the TTM. The TTM system was based on information extraction methods. A similar approach was used to calculate semantic accuracy in other work in NLG, including comparative system evaluation in the E2E Generation Challenge [Juraska et al., 2018; Dušek et al., 2020; Wiseman et al., 2017; Shen et al., 2019].

The TTM relied on a rule-based automatic aligner that tagged each output utterance with the attributes and values that it realized. It was also used to calculate the SER. The aligner, and the process to calculate the SER was covered in more detail in Section 4.4. The output of the aligner was then used by the TTM extractor to construct an MR that matches the (potentially errorful) utterance that was generated by the NLG. We referred to this MR as the "retrofit MR". For additional insight, we also reported the percentage of **semantically perfect outputs** (perfect%). These are outputs where the SER is 0 and there are no semantic errors. This measure is analogous to the Sentence Error Rate used in speech recognition.

**Retrofit MRs for Self-Training.** The TTM was critical for the novel self-training method described in Section 7.4.2. The retrofit MRs matched the (errorful) NLG output: when these MR/NLG output pairs combine attributes from both sources, they provided novel corrected examples to add back into training.

## 7.4 Results

We ran two sets of experiments. We first ran all of the NLG models described in Section 7.3.1 on the COM test set, and automatically calculate SER and perfect% as described in Section 4.4 and Section 7.3.2. We report these results in Section 7.4.1. Section 7.4.2 motivates and describes the self-training method and presents the results. The final models generated semantically perfect outputs 83% of the time.

### 7.4.1 Initial Model Results

| **Model** | **Training** | **Test** | SER | PERFECT | |
|---|---|---|---|---|---|
| | | | | N | % |
| NOSUP | E2E + NYC | COM | **.45** | 106 | 3.5% |
| GUIDE | E2E + NYC | COM | .66 | 15 | 0.5% |
| ATTR | E2E + NYC | COM | .46 | 167 | **5.5%** |
| BOOL | E2E + NYC | COM | **.45** | 86 | 2.8% |

Table 7.1: SER and perfect% on test for each model type on the test of 3040 MRs (COM) that combine attributes from both sources.

**Semantic Accuracy.** Table 7.1 summarizes the semantic results across the four models NOSUP, GUIDE, ATTR and BOOL. Overall, these results show that the task, and the COM test set, were surprisingly adversarial. All of the models had extremely high SER, and the SER for NOSUP, ATTR, and BOOL are very similar. Row 2 shows that the GUIDE model had much worse performance than the other models, in contrast to other tasks [Iyyer et al., 2018]. Because of this we did not examine the GUIDE model further. Row 3 shows that the ATTR supervision results in the largest percentage of perfect outputs (5.5%). This made it the best model of the three remaining models but there is much room for improvement since this is a very low number of correct outputs for an NNLG model.

| Model | Training | Test | SER | PERF % |
|---|---|---|---|---|
| NOSUP | E2E | E2E | .16 | 19% |
| NOSUP | E2E + NYC | E2E | .18 | 15% |
| NOSUP | NYC | NYC | .06 | 69% |
| NOSUP | E2E + NYC | NYC | .06 | 71% |

Table 7.2: Baseline results for each source on its own test using the NOSUP model. E2E test N = 630. NYC test N = 314.

In order to verify that these initial results are quite poor, the results in Table 7.1 should be compared with the baselines for testing NOSUP on **only** E2E or NYC in Table 7.2. Both the E2E and NYC test sets consist of unseen inputs, where E2E is the standard E2E generation challenge test [Dušek et al., 2020], and NYC consists of novel MRs with baseline attribute frequencies matching the training data.[2] Rows 1

[2]Previous work on the E2E dataset has also used seq2seq models, with SOA results for SER of 1% [Dušek et al., 2020] but here we do not use the full training set. The partition of the NYC dataset has not been used before but experiments on comparable NYC datasets have SERs of 0.06 and 0.02 [Reed et al., 2018; Harrison et al., 2019].

and 3 tested models trained on only E2E or only NYC, while Rows 2 and 4 tested the same trained NOSUP model used in Row 1 of Table 7.1 on the E2E and NYC test sets respectively. Comparing Rows 1 and 2 shows that training on the same combined data used in Table 7.1 slightly degrades performance on E2E. However, this SER is still considerably lower than the 0.45 SER for the NOSUP model tested on the COM test set, shown in the first row of Table 7.1. Row 4 shows that the NOSUP model trained on the combined data appears to actually improve performance on the NYC test. The perfect% goes up from 69% in Row 3 to 71% on Row 4. The SER of 0.06 shown in Row 4, should also be compared to the 0.45 SER reported for the NOSUP model in the first row of Table 7.1. Again, this indicated that there is significant room for improvement. These results taken together establish that the combined MRs in the COM test provide a very different challenge than the E2E and NYC unseen test inputs.

However, despite the poor performance of the initial models, we hypothesized that there may be enough good outputs to experiment with self-training. Since the original training data had no combined outputs, decoding may benefit from even small numbers of training items added back for self-training.

| **Model** | NAT. | COHER. | GRAMMAT. |
|---|---|---|---|
| NOSUP | 4.04 | 4.13 | 4.12 |
| ATTR | **4.11** | **4.25** | 4.14 |
| BOOL | 3.97 | 4.18 | **4.25** |
| AGREEMENT | .63 | .62 | .65 |

Table 7.3: Human Evaluation for NOSUP (N = 100) ATTR (N = 100) and BOOL (N = 86) for Naturalness, Semantic Coherence, and Grammaticality.

**Human Evaluation.** The automatic SER results provided insight into the semantic accuracy of the models but no assessment of other aspects of performance. We thus conducted a human evaluation on Mechanical Turk to qualitatively assess fluency, coherency and grammaticality, as described in Section 4.3. We used the automatic SER to select 100 semantically perfect references from the NOSUP and the ATTR models' test outputs, and the 86 perfect references from BOOL. We asked five Turkers to judge on a scale of 1 (worst) to 5 (best) whether the utterance is: (1) fluent and natural; (2) semantically coherent; and (3) grammatically well-formed. Table 7.3 reports the average score for these qualitative metrics as well as the Turker agreement, using the average Pearson correlation across the Turkers. The results show that the agreement among Turkers is high, and that all the models perform well but that the ATTR model outputs are the most natural and coherent, while the BOOL model outputs are the most grammatical.

### 7.4.2 Self-Training

In order to conduct self-training experiments, we needed perfect outputs that combine attributes from both sources to add back into training. These outputs must also be natural, coherent and grammatical but Table 7.3 shows that this is true of all the models. A key idea for the novel self-training method was that the TTM (Section 7.3.2) automatically produces "retrofit" corrected MRs that match the output texts of the NLG models. Thus we expected that we can construct more perfect outputs for self-training by using retrofitting than by just using the small number reported in Table 7.1. Here, we first analyse the outputs of the initial models to show that self-training is feasible, and then explain the method and present results.

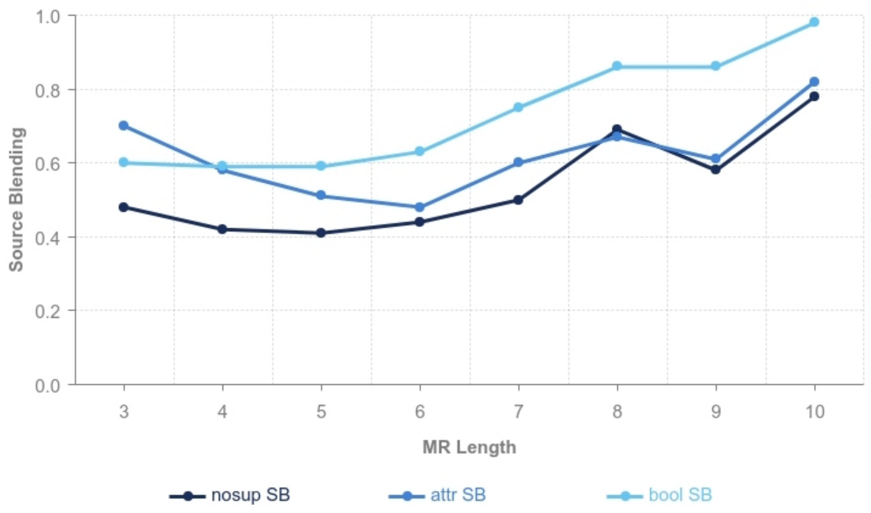


Figure 7.5: Source Blending Rate (SB) as a function of MR length for NOSUP, ATTR and BOOL.

**Error Analysis.** An initial examination of the outputs suggested that the models simply have trouble combining attributes from both sources. Table 7.4 provides examples of NLG model output utterances with high SERs. It illustrates how the NLG models struggle to combine attributes from the two ontologies which is required by all the input MRs (Column SB). It also illustrates cases where it is not possible to produce a valid retrofit MR that can be added back into training during self-training (Column Valid). In most cases these are due to many repetitions. Row 1 is an example where there is no source blending and since it has a repetition (*price*) it cannot be used for self-training (valid = no). Row 1 also illustrates an ungrammatical realization of *price* which we have no way to automatically detect at present "it is in the high price". Row 2 has three deletions as well as two repetitions. The output repeats "It is in midtown" three times in a row. Row 3 has five errors, it does not realize the dialogue act RECOMMEND and has deleted three other attributes and it hallucinations *food quality*. While this is a significant number of errors, this realization can still be used in self-training, since none of its errors are repetitions. Row 4 has all four types of errors. It deletes *cuisine*, *decor* and *service*, it realizes a value for *family-friendly* twice with different values, which is a substitution, and, finally, it hallucinates *food quality.* Row 5 actually has more errors than slots. It deletes all but two of its attributes: *name* and *rating.* It also hallucinates *food quality* and repeats *rating.*

To quantify this observation, we defined a metric, Source Blending Rate (**SB**), that counted the percentage of outputs that combine attributes from both sources, whether or not the attribute values are accurate:

$$SB = \frac{R_{sb}}{N}$$

where $R_{sb}$ is the count of references $r$ that contain an attribute $a_i \subseteq$ $source_1$ and another attribute $a_j \subseteq$ $source_2$, and N is the total number of references. Only attributes that appear uniquely in each source are included in the $a_i$, $a_j$: the unique attributes are illustrated in Figure 7.2.

Figure 7.5 graphs SB as a function of MR length. This shows that indeed the models **do** in many cases produce combined outputs and that the type of model supervision greatly influences SB. The NOSUP model was the worst: a fact that is masked by the NOSUP model's SER in Table 7.1, which appeared to be on a par with both ATTR and BOOL. Interestingly, all models were more likely to produce an SB output as the MRs get longer but Figure 7.5 shows clearly that the BOOL model especially excels.

For self-training, we also needed a model that generates utterances with the

| # | SER | Valid | SB | MR | Realization |
|---|---|---|---|---|---|
| 1 | .60 | no | no | name[RESTAURANT], `decor[fantastic]`, `qual[acceptable]`, price[high], *familyFriendly[yes]* | [RESTAURANT] is a child friendly restaurant with french food and it is in the high price range. it is in the high price. |
| 2 | .63 | no | no | name[RESTAURANT], `recommend[yes]`, cuisine[japanese], `decor[bad]`, location[midtown], `service[fantastic]`, *rating[low]*, *near[POINT-OF-INTEREST]* | [RESTAURANT] is the best restaurant since it is a japanese restaurant with bad ambiance and it is in midtown. it is in midtown. it is in midtown. |
| 3 | .71 | yes | yes | name[RESTAURANT], `recommend[yes]`, `decor[fantastic]`, `service[acceptable]`, *eatType[pub]*, *rating[high]*, *near[xnear]* | [RESTAURANT] is a pub with good food and it is located near the xnear. |
| 4 | .75 | no | yes | name[RESTAURANT], cuisine[southern], `decor[fantastic]`, location[city centre], price[cheap], `service[good]`, *eatType[coffee shop]*, *familyFriendly[no]* | [RESTAURANT] is a cheap, family friendly coffee shop with good food. it is in the city centre. it is not family friendly. |
| 5 | 1.17 | no | yes | name[RESTAURANT], cuisine[indian], `decor[fantastic]`, `service[good]`, *rating[high]*, *near[POINT-OF-INTEREST]* | [RESTAURANT] provides good food and has great customer rating and has great customer rating . |

.

Table 7.4: Example pathological outputs with high SERs from the NLG models before self-training. Valid realizations (col 3) are defined as those usable for self-training. In the MRs, the NYC attributes are represented using `typewriter font` and E2E attributes are represented using *italics*.

RECOMMEND dialogue act. As mentioned in Section 7.2.2, recommendations increased both semantic and syntactic complexity. Half the test items contained a recommendation, so we needed a model that can produce them. Table 7.5 presents results for SER and SB depending on whether a RECOMMEND was in the MR, showing that the three models vary a great deal. However, the BOOL row for the SB column shows that when the MR includes a recommendation, the BOOL model produces a combined output far more frequently than NOSUP or ATTR (SB = 0.73).

| **Model** | SER | | SB | |
|---|---|---|---|---|
| | REC | NO-REC | REC | NO-REC |
| NOSUP | .43 | .46 | .44 | .56 |
| ATTR | .51 | .41 | .36 | .77 |
| BOOL | .47 | .43 | .73 | .67 |

Table 7.5: Effect of the RECOMMEND dialogue act on Slot Error Rate (SER) and Source Blending (SB) for the three types of model supervision: NOSUP, ATTR and BOOL.

Thus Figure 7.5 and Table 7.5 show that the BOOL model produced the most combined outputs. After TTM extraction, the BOOL model provided the most instances (1405) of retrofit MR/output pairs to add to self-training. We therefore used BOOL in the self-training experiments below.

**Retrofitting MRs for Self-Training.** Table 7.6 illustrates how the TTM works, and shows that it can effectively create a new MR that may not have been previously seen in the training data. This allowed the model to **learn from its mistakes**. In Table 7.6 we can see two examples that start from different original MRs (col 1), but yield the same MR after text-to-MR extraction (col 2). In Row 1, the model output in column 3 deleted the attributes *price*, *decor* and *eat type* (pub), and, for *food quality*, substituted the value

| **Original MR** | **Text-to-MR** | **OUTPUT** |
|---|---|---|
| name[RESTAURANT], cuisine[fastfood], decor[good], qual[fantastic], location[riverside], price[cheap], eatType[pub], familyFriendly[no] | name[RESTAURANT], cuisine[fastfood], qual[good], location[riverside], familyFriendly[no] | [RESTAURANT] is a fast food restaurant located in the riverside area. it has good food and it is not family friendly. |
| name[RESTAURANT], recommend[yes], cuisine[fastfood], qual[good], location[riverside], familyFriendly[no] | name[RESTAURANT], cuisine[fastfood], qual[good], location[riverside], familyFriendly[no] | [RESTAURANT] is a fast food restaurant in the riverside area. it is not family friendly and has good food. |

Table 7.6: Examples to show retrofitting.

"good" for "fantastic". In Row 2, the model deleted the RECOMMEND dialogue act, but otherwise realized the original MR correctly. At test time, the original MRs produced different outputs (col 3). Thus the retrofitting yields two unique novel instances for self-training.

It is important to again note that the retrofit MRs for some NLG outputs **cannot** be used for self-training. NLG model outputs, whose semantic errors include repetitions, can **never** be used in self-training, because valid MRs do not include repeated attributes and values, and the method does not edit the NLG output string. However, deletion errors cause no issues: the retrofit MR simply does not have that attribute. Substitutions and hallucinations can also be used because the retrofit MR substitutes a value or adds an attribute and value to the MR. This could occur as long as the realized attribute and values were valid, e.g. "friendly food" is not a valid value for *food quality*.[3,4]

**Experiments.** To begin the self-training experiments, we applied the source-blending metric (SB) defined above to identify candidates that combine attributes from both sources. We then applied the TTM to construct MRs that match the NLG model outputs, as illustrated in Table 7.6, eliminating references that contain a repetition. We started with the same combined 76,832 training examples and the 1405 retrofit MR/NLG outputs from the BOOL model. We explored two bootstrapping regimes, depending on whether a model output is a repetition of one that we have already seen in the training data. One model kept repetitions and added them back into training, which we dubbed S-Repeat. The other model only added unique outputs back into training, which we dubbed S-Unique.

**Quantitative Results.** Figure 7.6 shows how the SER and perfect% continuously improved on the COM test set for S-Repeat over ten rounds of self-training, and that S-Repeat had better performance. This indicated that adding multiple instances of the same item to training was useful. The performance on the COM test set of the S-Unique model flattened after eight rounds. After ten rounds, the S-Repeat model had an SER of 0.03 and produced perfect outputs 82.9% of the time, a 77.4 percent absolute improvement over the best results in Table 7.1.

**COM-2 Test Set.** The self-training procedure used the COM test set during self-

[3]We applied the human evaluation in Section 7.3.2 to instances included in self-training: the correlation between human judgements and the automatic SER is 0.95, indicating that the retrofit MRs are highly accurate.

[4]Table 7.4 in provides additional examples of errorful outputs that **can** or **cannot** be used in self-training.

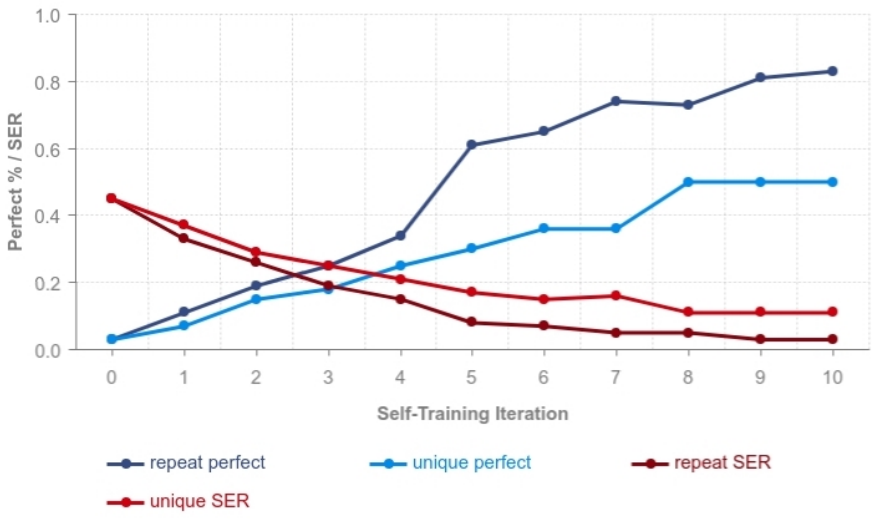


Figure 7.6: SER and perfect% on the COM test set for S-Repeat vs. S-Unique during self-training.

training and therefore potentially added the MRs from the COM test set back into training. To address this, we constructed a new test with 3040 novel MRs using the procedure described in Section 7.2.2, which we called COM-2. First we tested the initial models on COM-2, resulting in a best SER of 0.45 for the BOOL model, identical with the result for COM. For perfect% the best result was 5.3% on the ATTR model, which is again comparable to the original COM test set. This demonstrated that COM-2 was just as adversarial for the initial models as COM. We then tested the final self-trained model on COM-2, with the result that the SER for S-Repeat (0.03) and S-Unique (0.11) are again identical to the result for COM. The perfect% is comparable to that reported in Figure 7.6; it decreased by 2.2% for S-Repeat to 80.7% and increased by 0.2% for S-Unique to 50.7%. Overall, the performance on COM-2 improved by an absolute 75.4%, a significant improvement.

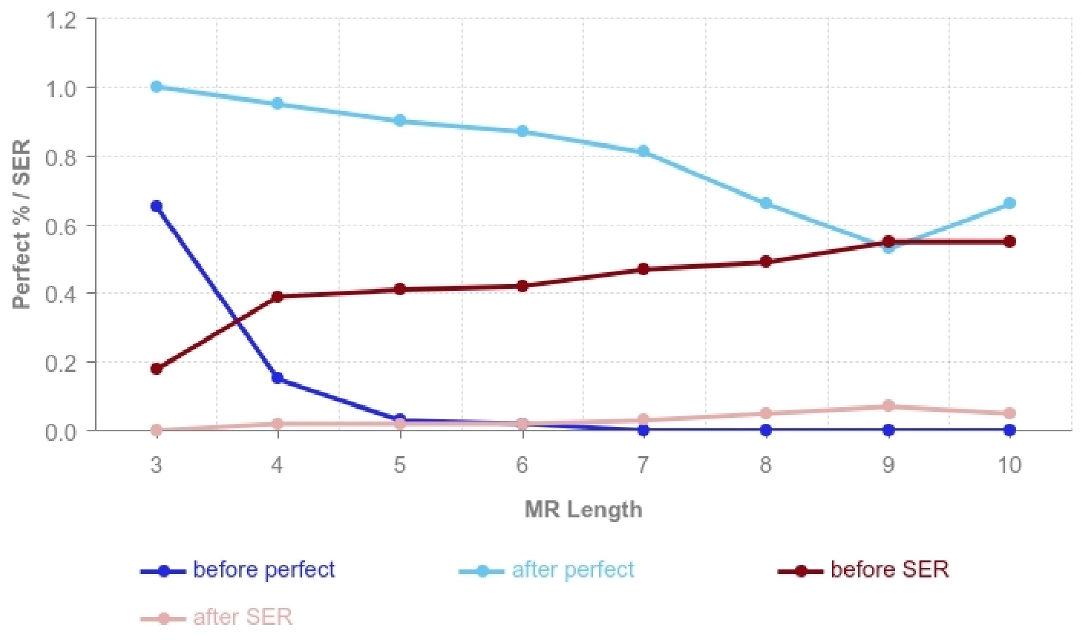


Figure 7.7: SER and perfect% on COM-2 as a function of MR length for BOOL supervision before self-training and for the S-Repeat model after self-training.

Figure 7.7 shows that the results improve, not only overall but also by MR length. It plots the SER and perfect% results, by MR length, for the BOOL model before and after self-training. While the perfect% decreased as the number of attributes increase, there is a large improvement over the initial model results. Also, after self-training the worst perfect% was still above 0.5, which was higher than perfect% for any MR length before self-training. The SER also improved over all MR lengths after self-training, not exceeding 0.06, significantly better than even the shortest MR before self-training.

**Performance on E2E and NYC test sets.** Table 7.2 provided a baseline for NOSUP's performance before self-training on the original test sets for E2E and NYC. We also verified that the self-trained model performs well after self-training. Table 7.7 shows that self-training improves the results for the original E2E and NYC test sets.

| **Model** | **Training** | **Test** | SER | PERF % |
|---|---|---|---|---|
| BOOL | S-REPEAT | E2E | .14 | 25% |
| BOOL | S-REPEAT | NYC | .05 | 77% |

Table 7.7: Performance of the self-trained S-Repeat model on the original E2E and NYC test sets. E2E test N = 630. NYC test N = 314.

**Human Evaluation.** We performed a human evaluation on Mechanical Turk to assess the qualitative properties of the model outputs after self-training. We selected 100 perfect references for S-Repeat and 100 for S-Unique and used the same HIT as in Section 7.4.1. Table 7.8 reports the average score for these qualitative metrics as well as the Turker agreement, using the average Pearson correlation, across the Turkers. The results show that naturalness, coherence and grammaticality are still high after self-training for both models, but that the S-Unique model produce better outputs from a qualitative perspective. We believed we could improve the self-training method used here with additional referenceless evaluation metrics that aim to measure naturalness and grammaticality [Mehri and Eskenazi, 2020].

| **Model** | NAT. | COHER. | GRAMMAT. |
|---|---|---|---|
| S-REPEAT | 3.99 | 4.08 | 4.02 |
| S-UNIQUE | **4.06** | **4.13** | **4.14** |
| AGREEMENT | .57 | .61 | .57 |

Table 7.8: Human Evaluation on Mechanical Turk for S-Repeat (N = 100) and S-Unique (N = 100) for Naturalness, Semantic Coherence, and Grammaticality.

**Qualitative and Linguistic Analysis.** Table 7.9 provides outputs from the models

| # | Realization |
|---|---|
| 1 | [RESTAURANT] is the best place because it is a family friendly pub with good decor and good food. |
| 2 | [RESTAURANT] is a family friendly restaurant with bland food and is in the low price range. It is the best restaurant. |
| 3 | [RESTAURANT] is a family friendly coffee shop with decent service and a low customer rating. It is in the £20-25 price range. |
| 4 | [RESTAURANT] is the best restaurant because it is in the east village, it is near [POINT-OF-INTEREST] with great service and it is affordable. |

Table 7.9: Example outputs with source blending. NYC attributes are represented using red and E2E attributes are represented using blue.

that display different ways of combining attributes from the original sources. In Row 1 we can see that the RECOMMEND dialogue act from NYC can be combined in the same sentence as the attributes *family-friendly* and *eat type* from E2E and aggregate these E2E attributes with NYC attributes *decor* and *food quality* using a "with" operator. Row 2 shows another example where the NYC and E2E attributes are joined using a "with" operator. In Row 3 there is a single sentence with four attributes where the NYC attribute is preceded and followed by E2E attributes. Row 4 concatenates the two sources in a single sentence using sentence coordination. The "east village" location from the NYC dataset, is concatenated with the attributes *near* from E2E and *service* from NYC. These examples show that the NLG models can combine attributes from both sources in many different ways.

Additional examples, in Table 7.10, provide outputs from the final iteration of self-training that display different ways of combining different attributes from the ontologies along with their input MRs. Row 1 shows that the model can combine attributes from the two sources in the same sentence, with attributes from each source, *decor* and *rating*, appearing in a single sentence with an "and" operation. Row 2 shows a different way of combining attributes from the two sources, with *family-friendly* and *food quality*, in a single sentence, this time using a "with" operation. In Row 3 we can see that the model can also generate complex sentences for recommendations using the "because" operation. Also, the attribute used in the *because* clause is from E2E, i.e. *family-friendly* but such sentences never appear in the original E2E training data. The last row shows a complex sentence where *decor* is combined with *eat type* and *customer rating*, again a novel combination.

| # | **MR** | **Realization** |
|---|---|---|
| 1 | name[RESTAURANT],cuisine[Indian], `decor[fantastic]`,`qual[good]`, `service[good]`, *rating[high]*, *near[POINT-OF-INTEREST]* | [RESTAURANT] is a Indian restaurant with great service and excellent ambiance and a high customer rating. It is located near the [POINT-OF-INTEREST]. |
| 2 | name[RESTAURANT], `recommend[yes]`, `decor[good]`, `qual[good]`, price[cheap], `service[bad]`, *familyFriendly[yes]*, | [RESTAURANT] is a family friendly restaurant with good food, good ambiance and bad service. It is in the low price range. |
| 3 | name[RESTAURANT], `recommend[yes]`, `decor[bad]`, `qual[good]`, location[flatiron/union square], price[cheap], `service[acceptable]`, *eatType[coffee shop]*, *rating[3 out of 5]*, | [RESTAURANT] is the best restaurant because it is a family friendly coffee shop with good food, friendly service and bad ambiance. It is in Flatiron/Union Square. It has a customer rating of 3 out of 5. It is cheap. |
| 4 | name[RESTAURANT], `recommend[yes]`, cuisine[mediterranean], `decor[fantastic]`, price[very expensive], *eatType[pub]*, *rating[5 out of 5]* | [RESTAURANT] is a Mediterranean pub with excellent ambiance and a customer rating of 5 out of 5. It is in the upper price range. It is the best restaurant. |

Table 7.10: Example outputs of source blending from final self-training iterations. In the MRs, the NYC attributes are represented using `typewriter font` and E2E attributes are represented using *italics*.

## 7.5 Pre-trained Model

Currently, massively pre-trained models are being tested for language tasks spanning from summarization to question answering to reading comprehension. These models, such as OpenAI's GPT-2 [Radford et al., 2019], Google's BERT [Devlin et al., 2019] and ELMo [Peters et al., 2018], are natural language models which are pre-trained on millions of pieces of texts from the internet. This allows for a model that has some working knowledge of language constructs before training on a specific task. Previously, models only had the specific task's dataset from which to learn natural language. There has also been success in using these models for open domain tasks such as question answering [Yang et al., 2019a; Alberti et al., 2019; Wang et al., 2019]. Because of this we decided to tune one of these models, OpenAI's GPT-2, with the dataset to see if a pre-trained model could improve the results.

### 7.5.1 GPT-2 Model

GPT-2 is a transformer-based language model created by OpenAI that has 1.5 billion parameters and is pre-trained on 8 million webpages [Radford et al., 2019]. Though the full model is not released to the public, we used a smaller model that is available on the GitHub.[5] GPT-2 was created to be able to predict the next word given the previous words in the text. GPT-2 used an unsupervised pre-training framework using a standard language modeling objective and a multi-layer Transformer decoder [Liu et al., 2018] based off the GPT model framework [Radford and Narasimhan, 2018].

[5]https://github.com/huggingface/transformers

Their training data focused on high quality data from many sources, using outbound Reddit links with at least three karma as a heuristic to only keep webpages which users found engaging. GPT-2 had state of the art performance on seven of the eight tested language modeling datasets, and confirmed that although there is some overlap between their training data and the data in their test sets, this n-gram overlap was not significantly larger than standard overlaps in the training and testing data, indicating that GPT-2 is not simply memorizing but instead is generalizing.

### 7.5.2 Training

While GPT-2 is a pre-trained model, the task cannot be handled by a zero-shot model. The MRs were a very particular type of data input and would not be represented in the data used for pre-training. This means that we needed to perform task-specific fine tuning before we could test how GPT-2 performed on the source blending task.

Since GPT-2 is a language model, and not a seq2seq model, we had to modify the training and test data. There is not a separate encoder and decoder to input the MR and the text to respectively. Instead, we created a single input with the MR and the text of the review concatenated together. We used an *end-token* to indicate where the MR ends and the text begins. Since GPT-2 learns the next word based on all previous words in the text, it first reads in the MR and uses that to inform what should appear in the output text. Because of this, the MR was handled as a string and was not transformed into a 1-hot vector encoding. At test time, we inputted just the MR and the *end-token*. We allowed the model to output 100 words to guarantee the model was not cut off. When processing the output of the model, we split the output text at the *end-token* and everything after that token was considered the review the GPT-2 model generated. We used gpt2-small with a batch size of 4 and half-precision floating point run on Google Colab.

We trained the model with three levels of supervision, NOSUP, BOOL and ATTR, as described in Section 7.3.1, where NOSUP has no additional supervision besides the attribute value pairs, BOOL has an additional token appended to the end of the MR which has the source described as a pair of booleans, False||True for NYC, True||False for E2E and True||True for the blended test, and ATTR has these boolean tokens appended to each attribute indicating its source as NYC, E2E or both.

### 7.5.3 Results

| **Model** | **Test** | SER | PERFECT | | SB N |
|---|---|---|---|---|---|
| | | | N | % | |
| NOSUP | COM | .46 | 139 | 4.6% | 1580 |
| ATTR | COM | .45 | 137 | 4.5% | 1408 |
| BOOL | COM | .45 | 139 | 4.6% | 1590 |

Table 7.11: SER, perfect% and SB N for initial GPT-2 fine tuned model for each type of supervision on the test of 3040 MRs (COM).

After training the GPT-2 model with NOSUP, ATTR and BOOL, we generated test output with the same test set, COM, from the original experiment. The results, seen in Table 7.11, for both SER and perfect% are very similar for all three types of supervision. In order to determine if GPT-2 improved the results from the initial findings using OpenNMT-py, we compared the results in Table 7.11 to the results in

Table 7.1. We found that both of these results are comparable. This indicated that using a pre-trained model does not actually improve the results. In order to confirm if GPT-2 leads to any improvements in this type of generalization, we needed to perform self-training, as in Section 7.4.2. We also confirmed that there are significant instances of source blending in the test output, which will allow us to perform the same self-training regime.

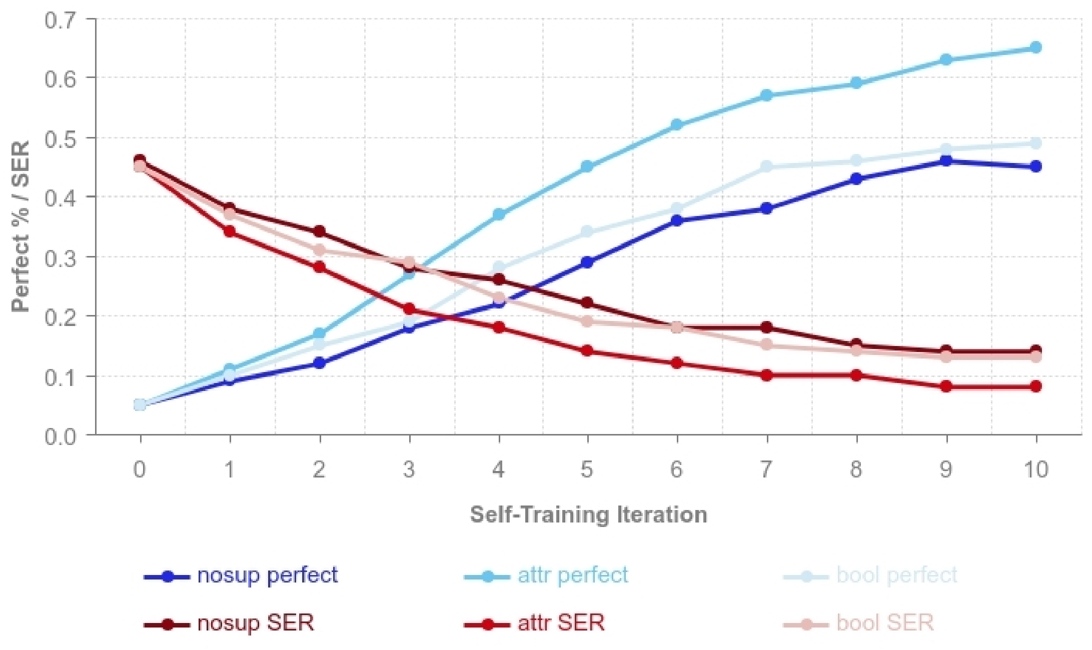


Figure 7.8: SER and perfect% on COM as a function of self-training iteration for NOSUP, BOOL and ATTR.

**Self-training Results.** Since all three types of supervision, NOSUP, BOOL and ATTR, had very similar results, we decided to attempt to do self-training with all three models. Even though we only performed self-training on BOOL previously, all three models here had very similar results, so we wanted to know if, while self-training, these results diverged or stayed similar. We followed the same self-training procedure described in Section 7.4.2 and we used the S-Repeat schema since it had better results than just using unique instances when previously self-training using the OpenNMT-py model.

In Table 7.8, we can see that the self-training does improve the results at each iteration across all three supervisions, though ATTR has a better perfect% than the other two supervisions. After ten iterations of self-training ATTR had an SER of 0.08 and a perfect% of 64.6%. This is better than the other two supervisions but it is still worse than the best results of the OpenNMT-py model, which had a best perfect% of 82.9%. This indicates that GPT-2 does not perform better than OpenNMT-py, though, we do observe that, unlike for OpenNMT-py, GPT-2 is still improving after ten iterations.

We continued to perform self-training for another ten iterations for ATTR (for a total of 20 iterations) and compared it to the final results with OpenNMT-py. In Table 7.12, we can see that OpenNMT-py and GPT-2's results were very similar and GPT-2 did not lead to any improvements of the semantic accuracy of the results. This was potentially because the OpenNMT-py model performed so well after self-training that it will be difficult to improve on them, based only on semantic evaluation. Merely using a pre-trained model does not make this task any easier to perform.

**Stylistic Analysis.** While there were not any improvements in terms of semantic accuracy when comparing OpenNMT-py to GPT-2, we were also curious if there might be any improvements in stylistic diversity. GPT-2, since it has been pre-trained on millions of utterances, might retain more stylistic features, which are often lost when performing self-training. This was because the model learns to generate with data it

| NNLG | Sup | Iter | SER | PERFECT | |
|---|---|---|---|---|---|
| | | | | N | % |
| OpenNMT-py | bool | 0 | .45 | 87 | 2.9% |
| OpenNMT-py | bool | 10 | .03 | 2520 | 82.9% |
| GPT-2 | attr | 0 | .45 | 137 | 4.5% |
| GPT-2 | attr | 20 | .04 | 2419 | 79.6% |

Table 7.12: SER and perfect% comparing the initial models to the final models for OpenNMT-py and GPT-2 on the test set COM.

has generated previously and can converge on only a small number of ways to generate output. GPT-2 was trained on a much larger vocabulary with many interesting features such as contrast.

We measured several stylistic features to compare OpenNMT-py and GPT-2. Since high semantic accuracy is important for models we only tested these results on the final iterations of self-training for both models. The first stylistic feature we measured was the size of the vocabulary, which is how many unique words are in the output. Then we measured the average sentence length, since longer sentences have more content per sentence and therefore have to aggregate together different attributes in ways other than separate sentences. We then also counted the number of times specific features appeared, the first number of utterances with a recommendation, then the number with contrast and finally the number with a relative clause. These statistics are in Table 7.13.

In the results, we can see that the stylistic variation is slightly better in the GPT-2 output. For both models, most features, other than recommendations, have fewer instances or decreased in stylistic diversity from before to after self-training. For OpenNMT-py, both the number of contrasts and the number of relative clauses decreases

| **NNLG** | **Sup** | **Iter** | **vocab N** | **av sent len** | **recom. N** | **contrast N** | **relative N** |
|---|---|---|---|---|---|---|---|
| OpenNMT-py | bool | 0 | 158 | 11.13 | 256 | 68 | 64 |
| OpenNMT-py | bool | 10 | 92 | 10.06 | 1506 | 0 | 0 |
| GPT-2 | attr | 0 | 422 | 11.34 | 730 | 205 | 191 |
| GPT-2 | attr | 20 | 213 | 9.51 | 1560 | 118 | 62 |

Table 7.13: SER and perfect% comparing the initial models' to the final models' stylistic diversity for OpenNMT-py and GPT-2 on the test set COM.

to 0. Recommendations is the one feature that increased, which is not surprising as it is a feature in the MR and therefore is controlled and encouraged to be generated by the model. It should increase as the semantic accuracy improves. For GPT-2, the vocabulary size started and remained higher than OpenNMT-py and the number of recommendations, contrasts and relative clauses also started and remained higher before and after self-training when compared to OpenNMT-py. Of the stylistic features measured, GPT-2 only performs worse than OpenNMT-py for average sentence length. Though it started higher before self-training, performing self-training caused a significant drop in sentence length and it was worse for GPT-2 than OpenNMT-py in the final iteration of self-training. Most significant was that GPT-2 after self-training still had contrast and relative clauses appear in its output, so it does appear that using a pre-trained model helped improve the stylistic variation of the output, though more work into this in particular is needed.

We also performed human evaluation on the output from the final iteration of self-training, using the same procedure described in Section 7.4.1. The results of this experiment are in Table 7.14. The final iteration of self-training for GPT-2 is worse than

| Model | Nat. | Coher. | Grammat. |
|---|---|---|---|
| GPT-2 | 3.73 | 3.63 | 3.77 |
| agreement | 0.48 | 0.49 | 0.49 |
| S-Repeat | 3.99 | 4.08 | 4.02 |
| S-Unique | 4.06 | 4.13 | 4.14 |
| agreement | 0.57 | 0.61 | 0.57 |

Table 7.14: Human Evaluation on Mechanical Turk for GPT-2 after 20 iterations of self-training for Naturalness, Semantic Coherence, and Grammaticality. Also, results from human analysis after self-training on OpenNMT-py are included for ease of comparison.

both S-Repeat and S-Unique self-training regimes using OpenNMT-py. This shows that even though the results are more stylistically varied, pre-training does not automatically lead to more natural, coherent or grammatical utterances.

## 7.6 Summary

Here, we present the first experiments on training an NLG for an extended domain ontology by re-using existing within-domain training data. We show that we can combine two training datasets for the restaurant domain, that have different ontologies, and generate output that combines attributes from both sources. We do this by applying a combination of neural supervision and a novel self-training method. While it is common practice to construct test sets with unseen attribute combinations, we know of no prior work based on constructing a new combined ontology. The experiments show that the task is surprisingly adversarial, consistent with recent work suggesting that neural models often fail to generalize [Wallace et al., 2019; Feng et al., 2018; Ribeiro et al., 2018; Goodfellow et al., 2014]. Work on domain transfer shares similar goals to the experiments presented here [Wen et al., 2016; Golovanov et al., 2019] but these methods do not produce NLG outputs that integrate attributes from two different sources into the same sentence.

In Section 7.2, we described a goal of this thesis, ontology merging, and how it can be performed. We take two separate ontologies, map similar attributes so they are represented using the same name, and create new MRs that use attributes and values from both sources. In this section we also described the training and testing datasets. We described the experimental overview, including the model architecture, in Section 7.3. We experimented with four types of supervision. The first is the baseline, NOSUP, where we just use the baseline architecture and no additional features in the MR. The second was ATTR, where two booleans representing the source ontology of the attribute was concatenated to every attribute in the MR. The third type of supervision was BOOL, where two booleans representing the source ontology of the entire utterance was added as a separate feature to the end of the MR. The final supervision was GUIDE, where the token concatenated to the end of the MR for BOOL was added directly to the decoder.

We presented the results in Section 7.4. The initial results showed that this task is very adversarial and that none of the supervisions had high semantic accuracy. After analyzing the output, we realized that there was ontology blending in the utterances output by the model, they just had semantic errors when compared to the input MR. By "retrofitting" new MRs which matched the output, we were able to perform a self-training regime. By adding all utterances which had attributes from both ontologies, and no semantic errors such as repetitions, we were able to have 77.4% absolute improvement of perfect outputs on the original test set, COM, and a 75.4% absolute improvement on a new test set, COM-2.

Finally, in Section 7.5, we reran the experiments with a pre-trained model, GPT-2. This state-of-the-art model has been shown to lead to improved generalizing results in other work [Chen et al., 2020a; Peng et al., 2020]. The experiments with it did not lead to any improvements in semantic accuracy, even after 20 iterations of self-training. We saw slight improvements in stylistic variation with this pre-trained model.

The final results show that the ability of the self-training method to automatically construct new training instances, results in high quality natural, coherent and grammatical outputs with high semantic accuracy. Also, that using a pre-trained model such as GPT-2 can help improve the stylistic diversity of the output.

# Chapter 8

# Conclusions

## 8.1 Overview

Generalization is needed in Neural Natural Language Generation (NNLG) in order to appropriately control the output being generated and to avoid the data bottleneck. Generalization in NNLG involves generating output which is similar, but not the same, as data seen in the training data and deriving general rules from particulars. We have accomplished both of these tasks in the course of this thesis, through multiple techniques to improve upon NNLG models' ability to learn general forms of the training data. By being able to expand on existing training data, we reduce the amount of new training data that needs to be created to generate new text. In this final chapter, we summarize the contributions, describe potential applications of the work and go over both the limitations and the future directions of the work.

## 8.2 Contributions

In Chapter 1, we defined the task of generalization and explained why it is necessary in NNLG. We outlined Natural Language Generation and neural models as well as reviewed the state-of-the-art in these fields and in generalization in Chapter 2. In Chapter 3, we described both the datasets from other sources and the datasets we generated that we employed in the experiments. The experiments required automatic and human evaluation, and we described the limitations of these metrics and the metrics that we used in Chapter 4. In Chapters 5-7, we went over the experiments on generalization. First, in Chapter 5, we generalized multiple features to a single personality label and learned to combine personalities only seen separately in the training data. Then in Chapter 6, we generalized from sentence planning operations seen in the training data to new data in test as well as transferring sentence planning operations between datasets. And finally, in Chapter 7, we trained a model on two distinct datasets in the same domain with different attributes and learned to generate output with attributes from bout sources. In this section we will further describe the contributions of this thesis.

### 8.2.1 Generalization

Over the course of this thesis, we developed methods to enable NNLG models learned to generalize. We found that these models need additional information to perform the generalization tasks that we were experimenting with. Here we will go over what these tasks were, and what we learned.

The first task was generalizing multiple specific stylistic features, which make up a complex personality, using a single token and learning to generate these multiple stylistic features together. We found that the model can learn to generate these personality features together, with high levels of correlation for pragmatic markers and aggregation operations between the gold data and the outputs of the model for token level supervision. But, a more fine grained supervision led to better results for both pragmatic markers and aggregation operations across all but one personality for each, agreeable for pragmatic markers and disagreeable for aggregation operations, demonstrating that the token level supervision does not capture all of the stylistic features accurately.

The next task was creating novel personalities not seen in the training data by learning to generalize across multiple different personalities. We showed that the model could learn to combine features not seen together in the training data and create these novel personalities. We observed that these personalities do not combine the features in even proportions in the new personalities, but instead the model learns features to different degrees that are not reflected in the original training data.

We demonstrated that the NNLG models can learn sentence planning operations and generalize them to values that the operation was not applied to in the training data. This applied to new meaning representations in sentence scoping, where the model could, with a 89% accuracy, generalize the number of sentences to complexity labels which could be used to generate data based on reading ability. The NNLG model could also apply sentence planning operation to new adjectives, which we tested using distributive aggregation where the model learned to distribute on the adjectives “low” and “average” in the training data, and with a single binary supervision token could learn to distribute on the adjective “high” with 98% accuracy. We also demonstrated that transfer learning could apply to discourse operations and that a small amount of a discourse operation in one dataset can be augmented with another dataset to improve the accuracy in generating that discourse operation.

Finally we showed that ontologies could be merged together from different datasets with a small number of overlapping attributes and with self-training. We demonstrated that this was a very challenging task for these models. Before self-training the best model only had 5.5% of the outputs which perfectly matched the semantics of the input meaning representation. Self-training, seeded on a very small number of outputs that combined the semantics from both input datasets, led to very large improvements in the accuracy of the model. After self-training the model had a 77.4% absolute improvement over those best results with a perfect output percentage of 82.9%. Even though there has been many successes in few-shot learning, even with pre-trained models like GPT-2 [Radford and Narasimhan, 2018], NNLG models do not excel at transferring semantics from one dataset to another. We showed that GPT-2 had the same problems that a seq2seq model had accomplishing this generalization task, needing self-training to improve the semantic results, though GPT-2 did result in higher levels of stylistic variation in the final model.

### 8.2.2 Datasets

We created datasets in the course of this thesis with which to experiment with generalization using NNLG models and have released these datasets to the wider community. Here we will describe these datasets.

The first datasets is the personality corpus, PersonageNLG.[1]. This is a corpus of 88K MR/utterance pairs of training data in the restaurant domain which were generated using PERSONAGE. The utterances vary in style according to five personality types, agreeable, disagreeable, conscientious, unconscientious, and extrovert. There are also 1,390 pairs of test data.

The second dataset is the sentence planning corpus.[2] This is a set of approximately 205K meaning representation to natural language utterance pairs which demonstrate discourse relations, such as contrast and justification, as well as multiple aggregation operations. This corpus is divided into three separate datasets:

1. NYC - 76,823 MR/utterance pairs used in the contrast experiment in Section 6.5 and the source blending experiment in Chapter 7. This dataset is described in detail in Section 3.2.

2. Distributive Train - 63,690 MR/utterance pairs with synthetically generated utterances with and without the distributive aggregation operation. Each MR in this dataset has both the price and rating attributes. This dataset was used in Section 6.4.

[1]https://nlds.soe.ucsc.edu/stylistic-variation-nlg
[2]https://nlds.soe.ucsc.edu/sentence-planning-NLG

3. Sentence Scoping Train - 64,442 MR/utterance pairs with synthetically generated utterances which were generated with varying numbers of sentences. This dataset was used in Section 6.3.

And, finally, a third dataset where we included an even number of training data instances from E2E and NYC, which we used in Chapter 7.[3] This dataset also included a novel test dataset of just MRs which combined attributes from both sources. Each MR in this test set had to have at least one attribute which was unique to each of the two sources. It could then include any number of additional attributes from either source and any number of additional attributes that belonged to both sources. This test set has 3040 unique MRs.

## 8.3 Applications

Being able to generalize within and between datasets has multiple applications, the most salient of which is the ability to generate new data without needing to generate a new dataset. This is necessary as the field of NLG continues to grow, since generating datasets is prohibitively expensive and time-consuming. Groups without a large amount of resources cannot generate a new datasets every time they need to include a new stylistic feature in their output. Being able to generate novel data without new datasets is necessary to keep the field equitable.

Generation of datasets with personality and other stylistic features is important to creating text curated towards individuals. In persuasive settings discourse structuring

[3]https://nlds.soe.ucsc.edu/source-blending-NLG

can be critical, for example in recommendations for restaurants, hotels, etc [Scott and de Souza, 1990; Moore and Paris, 1993]. Controlling a complex stylistic schema with a single token could be crucial to generating output which complements the person listening or reading without needing to tune many features.

Generating restaurant reviews also has the application of allowing people who might otherwise struggle to produce a natural language review the opportunity to voice their opinions on a restaurant in an engaging way. If one considers the values of the meaning representation as inputs into a form then one can insert their ratings and opinions about different features of a restaurant, generate multiple natural language outputs and select the one which expresses their feelings in a way that they are comfortable with.

## 8.4 Future Work

For future work, we are interested in multiple variations and more thorough analysis of these existing systems. Most of the experiments were done using synthetic or crowdsourced data. We are interested in attempting all of these tasks with natural data. As we mentioned in Chapter 3, natural data has the most stylistic variation and is more fluent and engaging than synthetic or even crowdsourced data. While this is the goal, this is a difficult task because natural datasets that contain the necessary information to perform these experiments do not necessarily exist, and they would require extensive annotations and filtering to create the precise datasets that are needed for some of these experiments. We also have multiple potential future work tasks for each of the three experiments that are in this thesis.

For generating new personalities as in the multivoice experiments (Section 5.3), we want to expand on these experiments by both combining more personality traits to create a greater variety of personalities as well as combining different types of personality traits. This is since the Big Five Personality Traits are actually all combined to form a full personality, with each personality trait as a spectrum that the personality is on. So, a single trait in a personality will have some amount of introversion or extroversion, it is not extroversion by itself. By combining more personalities we are closer to modeling a full personality instead of just one or two personality traits.

While the experiments in sentence planning (Chapter 6) resulted in models with high accuracy with only basic supervision, there are many other sentence planning operations and discourse relations that still can be tested. Some potential future experiments could include controlling which content is combined in sentences and other experiments to learn operations on new words, such as fronting. We also only tested merging contrast between datasets. We also also interested in transferring other discourse relations, such as recommendation and concessions.

For future work for the source blending experiments (Chapter 7), we want to try blending data from different domains as well as different sources. While not all domains will form coherent, combined, utterances, there are domains that could be referred to together, such as hotels and restaurants. People often want a hotel with restaurants nearby, and hotels often have restaurants within them, so being able to refer to both together could be advantageous. A dialogue system might also want to talk about many other topics together in a single turn and could be aided by a system that could blend ontologies from multiple sources, such as recommending a movie and movie theater as well as a restaurant to go to afterward, or talking about a baseball game and the weather together. This is a much harder task, because the utterances that would be combined would have multiple subjects and it would be more likely that the output would be semantically incoherent, but this is a task that has many potential applications and is an extension of this work.

In addition to combining different domains, combining datasets, within or across domains, which have no overlapping attributes is another extension of the source blending results. The self-training regime relies on a model that can generate some utterances that combine facts from both datasets, but if no combined utterances are generated can self-training still be seeded with hand created utterances? This is much more likely in dataset that have no attributes in common. We are interested, if these models can be seeded with hand written utterances which combine attributes, how many would be needed to start self-training? These are questions we are interested in answering in future work.

While we performed some preliminary evaluation of stylistic features in the source blending output, we found that the majority of stylistic features we lost in the self-training regime. We want to improve upon this because just generating the same basic templates when generating NNLG output is not sufficient to create useful output. We want the model to retain the interesting and varied utterances which are present in the training data.

Though we attempted to source blend with the pre-trained model GPT-2, there are new architectures being created with great frequency. We are interested in exploring different architectures, either by testing existing architectures or creating one, which will improve the models' ability to generalize, especially in the ontology blending experiment. We hope, that even though initial experiments using a pre-trained model did not improve the results, that with further experimentation, that this task can be accomplished with a pre-trained model.

# Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. (2016). Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Chris Alberti, Kenton Lee, and Michael Collins (2019). A bert baseline for the natural questions.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba (2019). Constrained decoding for neural nlg from compositional representations in task-oriented dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 831–844.

Satanjeev Banerjee and Alon Lavie (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl work-*

*shop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Regina Barzilay and Mirella Lapata (2006). Aggregation via set partitioning for natural language generation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 359–366. Association for Computational Linguistics.

Anja Belz, Simon Mille, and David M Howcroft (2020). Disentangling the properties of human evaluation methods: A classification system to support comparability, meta-evaluation and reproducibility testing. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 183–194.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei (2020). Language models are few-shot learners.

Lynne Cahill, John Carroll, Roger Evans, Daniel Paiva, Richard Power, Donia Scott, and Kees van Deemter (2001). From rags to riches: exploiting the potential of a flexible generation architecture. In *Meeting of the Association for Computational Linguistics*.

Ernie Chang, Xiaoyu Shen, Dawei Zhu, Vera Demberg, and Hui Su (2021). Neural data-to-text generation with lm-based text augmentation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 758–768.

David L Chen and Raymond J Mooney (2008). Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135.

Wenhu Chen, Yu Su, Xifeng Yan, and William Yang Wang (2020a). Kgpt: Knowledge-grounded pre-training for data-to-text generation. *arXiv preprint arXiv:2010.02307*.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang (2020b). Few-shot nlg with pre-trained language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 183–190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

George Doddington (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145.

Ondřej Dušek (2017). Novel methods for natural language generation in spoken dialogue systems.

Ondrej Dusek and Filip Jurcícek (2016a). A context-aware natural language generator for dialogue systems. volume abs/1608.07076.

Ondrej Dusek and Filip Jurcícek (2016b). Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. volume abs/1606.05491.

Ondrej Dusek, Jekaterina Novikova, and Verena Rieser (2017). Referenceless quality estimation for natural language generation. *CoRR*, abs/1708.01759.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser (2020). Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computer Speech & Language*, 59:123–156.

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent (2010). Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings.

Angela Fan, David Grangier, and Michael Auli (2018). Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54.

Shi Feng, Eric Wallace, Pedro Grissom II, Alvin Rodriguez, Mohit Iyyer, and Jordan Boyd-Graber (2018). Pathologies of neural models make interpretation difficult. In *Empirical Methods in Natural Language Processing*.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer (2019). Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 552–562.

Jessica Ficler and Yoav Goldberg (2017). Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633*.

Tsu-Jui Fu, Peng-Hsuan Li, and Wei-Yun Ma (2019). Graphrel: Modeling text as relational graphs for joint entity and relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1409–1418.

Cristina Garbacea and Qiaozhu Mei (2020). Neural language generation: Formulation, methods, and evaluation.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini (2017). Creating training corpora for micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics.

Albert Gatt and Emiel Krahmer (2018). Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Xavier Glorot and Yoshua Bengio (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Lewis R Goldberg (1990). An alternative "description of personality": the big-five factor structure. *Journal of personality and social psychology*, 59(6):1216.

Sergey Golovanov, Rauf Kurbanov, Sergey Nikolenko, Kyryl Truskovskyi, Alexander Tselousov, and Thomas Wolf (2019). Large-scale transfer learning for natural language generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6053–6058.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Travis R Goodwin, Max E Savery, and Dina Demner-Fushman (2020). Towards zero-shot conditional summarization with adaptive multi-task fine-tuning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2020, page 3215. NIH Public Access.

Samuel D Gosling, Peter J Rentfrow, and William B Swann Jr (2003). A very brief measure of the big-five personality domains. *Journal of Research in personality*, 37(6):504–528.

Alex Graves (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850.*

Vrindavan Harrison, Lena Reed, Shereen Oraby, and Marilyn Walker (2019). Maximizing stylistic control and semantic accuracy in nlg: Personality variation and discourse contrast. In *Proceedings of the 1st Workshop on Discourse Structure in Neural NLG*, pages 1–12.

Vrindavan Harrison and Marilyn Walker (2018). Neural generation of diverse questions using answer focus, contextual and linguistic features. *11th International Conference on Natural Language Generation (INLG 2018).*

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen (2020). Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654.*

Sepp Hochreiter and Jürgen Schmidhuber (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

David M Howcroft, Anja Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser (2020). Twenty years of confusion in human evaluation: Nlg needs evaluation sheets and standardised definitions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182.

David M Howcroft, Dietrich Klakow, and Vera Demberg (2017). The extended sparky restaurant corpus: designing a corpus with variable information density. *Proc. Interspeech 2017*, pages 3757–3761.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing (2017). Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596. PMLR.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer (2018). Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885.

Nan Jiang, Jing Chen, Ri-Gui Zhou, Changxing Wu, Honglong Chen, Jiaqi Zheng, and Tao Wan (2020). Pan: Pipeline assisted neural networks model for data-to-text generation in social internet of things. *Information Sciences*, 530:167–179.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. (2017). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Juraj Juraska, Panagiotis Karagiannis, Kevin Bowden, and Marilyn Walker (2018). A deep ensemble model with slot alignment for sequence-to-sequence natural language generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 152–162.

Juraj Juraska and Marilyn Walker (2018). Characterizing variation in crowd-sourced data for training neural language generators to produce stylistically varied outputs. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 441–450.

Katharina Kann, Sascha Rothe, and Katja Filippova (2018). Sentence-level fluency evaluation: References help, but can be spared! In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 313–323.

Akbar Karimi, Leonardo Rossi, Andrea Prati, and Katharina Full (2020). Adversarial training for aspect-based sentiment analysis with bert. *arXiv preprint arXiv:2001.11316.*

Chris Kedzie and Kathleen McKeown (2019). A good sample is hard to find: Noise injection sampling and self-training for neural language generation models. In *Pro-*

*ceedings of the 12th International Conference on Natural Language Generation*, pages 584–593.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush (2017). Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72.

Karen Kukich (1983). Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 145–150.

Gerasimos Lampouras and Andreas Vlachos (2016). Imitation learning for language generation from unaligned data. In Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad, editors, *COLING*, pages 1101–1112. ACL.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer (2020). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Xinlong Li, Xingyu Fu, Guangluan Xu, Yang Yang, Jiuniu Wang, Li Jin, Qing Liu, and Tianyuan Xiang (2020). Enhancing bert representation with context-aware embedding for aspect-based sentiment analysis. *IEEE Access*, 8:46868–46876.

Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang (2020). Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1054–1064.

Chin-Yew Lin and Franz Josef Och (2004). Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 605–612.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer (2018). Generating wikipedia by summarizing long sequences.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Xiaofei Lu (2014). *Computational methods for corpus annotation and analysis*. Springer.

Yao Lu, Yue Dong, and Laurent Charlin (2020). Multi-xscience: A large-scale dataset for extreme multi-document summarization of scientific articles. *arXiv preprint arXiv:2010.14235*.

Thang Luong, Hieu Pham, and Christopher D Manning (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Computing Machinery (1950). Computing machinery and intelligence-am turing. *Mind*, 59(236):433.

François Mairesse and Marilyn Walker (2007). Personage: Personality generation for dialogue. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 496–503.

F. Mairesse and M.A. Walker (2010). Towards personality-based user adaptation: psychologically informed stylistic language generation. *User Modeling and User-Adapted Interaction*, pages 1–52.

Francois Mairesse and Marilyn A. Walker (2011). Controlling user perceptions of linguistic style: Trainable generation of personality traits. *Computational Linguistics.*

Nitika Mathur, Timothy Baldwin, and Trevor Cohn (2020). Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4984–4997, Online. Association for Computational Linguistics.

Shikib Mehri and Maxine Eskenazi (2020). Unsupervised evaluation of interactive dialog with dialogpt. In *Proc. of the SIGDIAL 2020.*

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter (2015). What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *CoRR*, abs/1509.00838.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

J. D. Moore and C. L. Paris (1993). Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*, 19(4).

Crystal Nakatsu (2008). Learning contrastive connectives in sentence realization ranking. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, pages 76–79. Association for Computational Linguistics.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, et al. (2020). Dart: Open-domain structured data record to text generation. *arXiv preprint arXiv:2007.02871*.

Neha Nayak, Dilek Hakkani-Tur, Marilyn Walker, and Larry Heck (2017). To plan or not to plan? discourse planning in slot-value informed sequence to sequence models for language generation. In *Proc. of Interspeech 2017*.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser (2017a). Why we need new evaluation metrics for nlg. *arXiv preprint arXiv:1707.06875*.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser (2017b). The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual Meeting* *of the Special Interest Group on Discourse and Dialogue*, Saarbrücken, Germany. arXiv:1706.09254.

Jekaterina Novikova, Oliver Lemon, and Verena Rieser (2016). Crowd-sourcing nlg data: Pictures elicit better data. In *Proceedings of the 9th International Natural Language Generation conference*, pages 265–273.

Shereen Oraby, Vrindavan Harrison, Abteen Ebrahimi, and Marilyn Walker (2019). Curate and generate: A corpus and method for joint control of semantics and style in neural nlg.

Shereen Oraby, Lena Reed, TS Sharath, Shubhangi Tandon, and Marilyn Walker (2018a). Neural multivoice models for expressing novel personalities in dialog. *Proc. Interspeech 2018*, pages 3057–3061.

Shereen Oraby, Lena Reed, Shubhangi Tandon, TS Sharath, Stephanie Lukin, and Marilyn Walker (2018b). Controlling personality-based stylistic variation with neural natural language generators. *Proceedings of the 19th SIGdial Workshop on Discourse and Dialogue.*

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das (2020). Totto: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186.

Cecile Paris (2015). *User modelling in text generation.* Bloomsbury Publishing.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Dean Peabody and Lewis R Goldberg (1989). Some determinants of factor structures from personality-trait descriptors. *Journal of personality and social psychology*, 57(3):552.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao (2020). Few-shot natural language generation for task-oriented dialog. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 172–182.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (2018). Deep contextualized word representations.

Vassilis Plachouras, Charese Smiley, Hiroko Bretz, Ola Taylor, Jochen L Leidner, Dezhao Song, and Frank Schilder (2016). Interacting with financial data using natural language. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1121–1124.

Ratish Puduppully, Li Dong, and Mirella Lapata (2019). Data-to-text generation with content selection and planning.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang (2020). Pre-trained models for natural language processing: A survey.

A. Radford and Karthik Narasimhan (2018). Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever (2019). Language Models are Unsupervised Multitask Learners.

O. Rambow, M. Rogati, and M. Walker (2001). Evaluating a trainable sentence planner for a spoken dialogue travel system. In *Proc. of the Meeting of the Association for Computational Lingustics, ACL 2001.*

Clément Rebuffel, Marco Roberti, Laure Soulier, Geoffrey Scoutheeten, Rossella Cancelliere, and Patrick Gallinari (2021). Controlling hallucinations at word level in data-to-text generation. *arXiv preprint arXiv:2102.02810.*

Lena Reed, Vrindavan Harrison, Shereen Oraby, Dilek Hakkani-Tur, and Marilyn Walker

(2020). Learning from mistakes: Combining ontologies via self-training for dialogue generation.

Lena Reed, Shereen Oraby, and Marilyn Walker (2018). Can neural generators for dialogue learn sentence planning and discourse structuring? In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 284–295.

Ehud Reiter and R. Dale (1997). Building applied natural language generation systems. *Nat. Lang. Eng.*, 3:57–87.

Ehud Reiter and Robert Dale (2000). *Building Natural Language Generation Systems.* Cambridge University Press.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin (2018). Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Ananya B Sai, Akash Kumar Mohankumar, and Mitesh M Khapra (2020). A survey of evaluation metrics used for nlg systems. *arXiv preprint arXiv:2008.12009.*

Donia R. Scott and Clarisse Sieckenius de Souza (1990). Getting the message across in RST-based text generation. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation.* Academic Press, London.

Thibault Sellam, Dipanjan Das, and Ankur Parikh (2020). BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch (2016). Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page 35–40. Association for Computational Linguistics.

Claude Elwood Shannon (2001). A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55.

James Shaw (1998). Clause aggregation using linguistic knowledge. In *Proc. of the 8th International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Ontario.

Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein (2019). Pragmatically informative text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4060–4067.

Xiaoyu Shen, Ernie Chang, Hui Su, Cheng Niu, and Dietrich Klakow (2020). Neural data-to-text generation via jointly learning the segmentation and correspondence. In

*Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7155–7165.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown (2004). Syntactic simplification for improving content selection in multi-document summarization. In *Proceedings of the 20th international conference on Computational Linguistics*, page 896. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Amanda Stent (2002). A conversation acts model for generating spoken dialogue contributions. *Computer Speech and Language: Special Issue on Spoken Language Generation.*

Amanda Stent, Matthew Marge, and Mohit Singhai (2005). Evaluating evaluation methods for generation in the presence of variation. In *Computational Linguistics and Intelligent Text Processing: 6th International Conference, CICLing 2005, Mexico City, Mexico, February 13-19, 2005, Proceedings*, volume 3406, page 341. Springer.

Amanda Stent, Rashmi Prasad, and Marilyn Walker (2004). Trainable sentence planning for complex information presentation in spoken dialogue systems. In *Meeting of the Association for Computational Linguistics.*

Amanda Stent, Marilyn Walker, Steve Whittaker, and Preetam Maloor (2002). User-tailored generation for spoken dialogue: An experiment. In *ICSLP*.

Elior Sulem, Omri Abend, and Ari Rappoport (2018). Bleu is not suitable for the evaluation of text simplification. *arXiv preprint arXiv:1810.05995*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Wilson L Taylor (1953). “cloze procedure”: A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.

Ye Tian, Ioannis Douratsos, and Isabel Groves (2018). Treat the system like a human student: Automatic naturalness evaluation of generated text without reference texts. *INLG 2018*, page 109.

Marilyn Walker, Owen Rambow, and Monica Rogati (2002a). Training a sentence planner for spoken dialogue using boosting. *Computer Speech and Language: Special Issue on Spoken Language Generation*, 16(3-4):409–433.

Marilyn A Walker, Steve Whittaker, Amanda Stent, Preetam Maloor, Johanna D Moore, Michael Johnston, and Gunaranjan Vasireddy (2002b). Speech-plans: Generating evaluative responses in spoken dialogue. In *Proceedings of the International Natural Language Generation Conference*, pages 73–80.

Marilyn A Walker, Stephen J Whittaker, Amanda Stent, Preetam Maloor, Johanna

Moore, Michael Johnston, and Gunaranjan Vasireddy (2004). Generation and evaluation of user tailored responses in multimodal dialogue. *Cognitive Science*, 28(5):811–840.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh (2019). Universal adversarial triggers for attacking and analyzing nlp. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang (2019). Multi-passage bert: A globally normalized bert model for open-domain question answering.

Leo Wanner, Harald Bosch, Nadjet Bouayad-Agha, Gerard Casamayor, Thomas Ertl, Désirée Hilbring, Lasse Johansson, Kostas Karatzas, Ari Karppinen, Ioannis Kompatsiaris, et al. (2015). Getting the environmental information across: from the web to the user. *Expert Systems*, 32(3):405–432.

Zeerak Waseem and Dirk Hovy (2016). Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-hao Su, David Vandyke, and Steve J. Young (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *CoRR*, abs/1508.01745.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young (2016). Multi-domain neural network language generation for spoken dialogue systems. *arXiv preprint arXiv:1603.01232*.

Sam Wiseman, Stuart Shieber, and Alexander Rush (2017). Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

Yuk Wah Wong and Raymond Mooney (2007). Generation by inverting a semantic parser that uses statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 172–179.

Hayahide Yamagishi, Shin Kanouchi, Takayuki Sato, and Mamoru Komachi (2016). Controlling the voice of a sentence in japanese-to-english neural machine translation. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, pages 203–210.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin (2019a). End-to-end open-domain question answering with. *Proceedings of the 2019 Conference of the North*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le (2019b). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 32:5753–5763.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao (2018). Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 506–514.

Ying Zhang, Stephan Vogel, and Alex Waibel (2004). Interpreting bleu/nist scores: How much improvement do we need to have a better system? In *LREC*.

Tiancheng Zhao and Maxine Eskenazi (2018). Zero-shot dialog generation with cross-domain latent actions. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 1–10.