$Vocabulary_{source}$, the seq2seq model aimed to predict a sequence $\{y_1, .., y_i\}$ where $y \in Vocabulary_{target}$. Seq2Seq models were first used for machine translation tasks where inputs from one language was translated to another language. Given the flexibility of the architecture, the model was applied to many other domains and use cases including program synthesis.

Current approaches to If-Then program synthesis exploit the formulaic nature of the IFTTT and Zapier recipes. They represent each component of the recipe as distinct multi-class labels. A downside to this is approach is if the recipe format is expanded, additional classification heads must be added. Additionally, existing models cannot predict more complicated recipes that may contain multiple channels, complex control structures and embedded function arguments. Seq2seq models offer a higher degree of flexibly in representing complex programs as sequences. We investigated if seq2seq could be successful in If-Then program synthesis, especially around predicting the recipe in its entirety given variable inputs.

4 Experiment

We evaluated three different seq2seq architectures: a baseline LSTM Encoder-Decoder, OpenNMT, and basic Transformer. We intentionally selected OpenNMT and the Transformer architectures given their recent popularity and general effectiveness in neural machine translation tasks. We describe the architecture in further detail below.

Both the IFTTT and Zapier datasets were preprocessed identically. Recipe titles were converted to lowercase. In the Zapier dataset, we found descriptions were high quality and improved all the performance of all models by 1 percent. Description were concatenated to the title using the "[SEP]" token.

The recipes were converted to a single space delimited sequence. If the channel or function consisted of multiple words, they were joined using underscores. Additionally, the channel was prefixed to the function. For example the following IFTTT recipe:

Trigger Channel: NY Times
Trigger Function: New Article Posted
Action Channel: Twitter
Action Function: New Post

was converted to the following sequence: *ny_times ny_times.new_article_posted twitter twitter.new_post*.

4.1 LSTM Encoder-Decoder

Our baseline was a simple LSTM Encoder-Decoder model. The model was implemented and trained using the JoeyNMT framework (**?**). The encoder and decoder are bidirectional LSTMs with multiplicative attention (**?**). The embedding size was 16, the hidden size was 64, and the dropout factor was .10. Inputs to the encoder were right padded tokenized word ids with a fixed sequence length of 25. All words in the input and target sequence vocabularies were used.

	LSTM	**OpenNMT**	**Transformer**
Sequence	84.75	93.23	**93.91**
Positional	94.39	97.66	**97.94**
Event Channel	97.18	99.09	**99.26**
Event Function	88.97	94.96	**95.47**
Action Channel	97.65	98.97	**99.16**
Action Function	93.74	97.61	**98.00**

Table 2: Seq2Seq model results on predicting Zapier recipes.

	LSTM	**OpenNMT**	**Trans.**	**Chen '16**	**Quirk '16**
Sequence	52.35	*55.41*	53.06	N/A	N/A
Positional	75.31	*76.39*	75.33	N/A	N/A
Trigger Channel	75.09	77.44	74.91	**91.60**	89.1
Trigger Function	61.37	63.72	62.27	**87.50**	71.00
Action Channel	85.02	84.66	84.12	**91.60**	89.10
Action Function	79.78	79.78	79.60	**87.50**	71.00

Table 3: Model performance on IFTTT recipes.

The model was trained on 100 epochs and evaluated every 4,000 steps. We use the adam optimizer with a learning rate of .001. Finally, we used cross-entropy as our loss function. The full configuration details for our baseline model can be found on our github repository.

4.2 OpenNMT

The second model we evaluated was the stacked RNN OpenNMT model provided in OpenNMT-py library (**?**). We used the default model and training script.They use a two layer RNN encoder and two layer stacked LSTM decoder. Global attention is added to the bottom. Inputs and target sequences are stored in separate files (segmented into training and validation sets) and provided to the training script. The model trained for 100,000 steps. Please see **?** for further model specifics.

4.3 Basic Transformer

The final model we evaluated was the Transformer model introduced in **? ?**. We used the JoeyNMT framework to implement and train the Transformer model. The model consisted of six layer transfor of the architecturemer encoder and decoder with eight attention heads and used additive attention (**?**). The input embedding size was 512, the hidden size was 512, the feed forward size 2048 and it had dropout factor of .10. We constrained the input vocabulary to the 4000 most frequently occurring word tokens. Inputs to the encoder were right padded tokenized word ids with a fixed sequence length of 30.

The model was trained on 100 epoch and validated every 4,000 steps. We used the noam learning rate scheduler (**?**) with learning rate factor of 1 and learn rate warmup value of 4,000. We used cross-entropy for our loss function. The full configuration details for this model can be found in our github repository.

5 Evaluation Metrics

In traditional language translations tasks BLEU and Rouge scores are often used as the primary evaluation metric. These

scores try to account for the fact that translations may vary in word usage and syntax from the source sequences. In our task, the predicted recipes must be correct across the channels and functions. Additionally, the order of predicted sequence matters as each position in sequence corresponds to a specific recipe component.

In addition to the recipe component accuracy scores (e.g. Trigger Channel accuracy or Event Function accuracy), we consider the overall sequence accuracy and the positional accuracy of the predicted sequences. The sequence accuracy measures if the overall sequence is correct. The prediction is assigned a score of 1 only if the predicted string exactly matches the reference string. We expect the sequence score to be lower in general as the predictions have a higher level of scrutiny. The positional score provides partial credit to the prediction, if the tokens correctly match the reference tokens at their position in the recipe. For example, if a prediction correctly identifies 3 of the 4 recipes components, it will receive a score of .75. We provide the distribution of positional errors in Table 4.

6 Results

We were able to successfully train all three architectures and had varying levels of success. In Table 3 we see that none of the seq2seq models were able to match the reported channel/function accuracy values in **? ?** and **? ?**. Both previous works only reported aggregate channel and function accuracy values. For simplicity, we assumed those aggregate values were the same across channels and respectively. Of the seq2seq models, OpenNMT consistently performed better than our baseline and the Transformer model. The seq2seq models did fairly well on Action Channel and Action function prediction, coming within 10 points of the **? ?** scores. We believe this may be related to how recipes are described in general. In both IFTTT and Zapier, users are more explicit in describing the action scenario (the "Then") part of the program than they are in describing the Triggering event.

In contrast to IFTTT, the seq2seq models all performed strongly on the Zapier dataset (Table 2). Our baseline LSTM encoder-decoder that had an overall sequence accuracy or 84.75% and strong performance on predicting each of the component recipes parts in context of the full recipe. The Transformer model performs the best is 9 points better that the baseline model in sequence accuracy. We significant improvement over Event Function (6 point increase) and Action Function (4 point increase) identification. Over we found the quality of the Zapier titles and descriptions better than those of IFTTT. Zapier titles were consistent their ability to concisely describe the recipes and the descriptions were also well written and provided useful clues

7 Discussion

Overall the seq2seq models were able to learn If-Then recipes in their entirety. The models were surprisingly robust and able to capture various linguistic variations and ambiguities found the natural language descriptions. Across both datasets, the models tended to have higher Action and Action Function scores than Trigger/Event scores. From superficial analysis, we found that users on average were more specific in describing the action then they were describing the triggering event. We had significant difficulty working with the IFTTT dataset. Nearly 50% of the train and validation sets we thrown away, and nearly 90% of the test set. Through visual and ad-hoc analysis, we would often find mistakes in both the annotated test set recipes and the scraped recipes. Given the age and volatility of the recipe urls, we were not confident that our experiment conditions matched those from previous works. We were unable to reproduce any of the prior results and therefore had difficulty doing a more through error analysis on our findings.

	Zapier			IFTTT		
Errors	**LSTM**	**ONMT**	**Trans.**	**LSTM**	**ONMT**	**Trans.**
Zero	84.75	93.23	9.39E-01	52.35	55.42	53.07
One	10.54	5.00	4.73E-02	14.44	12.09	13.36
Two	3.13	12.6	9.26E-03	23.10	21.84	22.92
Three	0.68	0.21	6.17E-04	2.35	3.97	2.71
Four	0.91	0.31	3.70E-03	7.76	6.68	7.94

Table 4: Distribution of errors across all predictions by domain.

In Table 4, we provide the distribution across all the predictions on the respective tests sets. It is interesting to note that on the IFTTT dataset, the model likely to make two errors as opposed to one, three or four. We believe there is a tight coupling between trigger channel and trigger function predictions. We hypothesize that if the model fails to predict the Trigger, it will also fail to predict the Trigger Functions.

The seq2seq model's performance on the Zapier dataset was very encouraging. As a next step we plan to investigate argument extraction on the dataset. Additionally, we are interested in explore the transfer learning potential given quality disparity between Zapier and IFTTT. Both have similiar vocabularies and recipe domains. We hypothesize a model trained on Zapier data may preform better on the IFTTT dataset than the model trained solely on IFTTT data. Finally, we are interested in investigating more complex program synthesis challenges. We believe more complex program representations can potentially be learned through seq2seq models.

8 Conclusion

In this paper we proposed modeling If-Then program synthesis as sequence learning task. The models attempted to learn how translate natural language descriptions into IFTTT and Zapier recipes. Three seq2seq architectures were evaluated: a baseline LSTM encoder-decoder, the OpenNMT Stacked RNN, and the Transformer model. The models were successfully trained and able to predict the full automation recipes in an end-to-end manner. Due to several challenges with IFTTT dataset, we found the seq2seq model performance to be adequate but unable to match the accuracy scores of prior work. Of the seq2seq models, the OpenNMT model performed the on the IFTTT dataset, with a sequence accuracy score of 55%, positional accuracy of 75 %, and overall Action Channel accuracy score of 79%. In contrast, the seq2seq models performed strongly on the Zapier dataset. The Transformer model score the highest across all