# Evaluating Seq2Seq Learning Models for If-Then Program Synthesis

**Dhairya Dalal and Byron V. Galbraith**
Talla, Inc
{dhairya, byron}@talla.com
2 Park Plaza, 7th Floor, Boston, MA 02116

## Abstract

Implementing enterprise process automation often requires significant technical expertise and engineering effort. It would be beneficial for non-technical users to be able to describe a business process in natural language and have an intelligent system generate the workflow that can be automatically executed. A building block of process automations are If-Then programs. In the consumer space, sites like IFTTT and Zapier allow users to create automations by defining If-Then programs using a graphical interface. We explore the efficacy of modeling If-Then programs as a sequence learning task. We find Seq2Seq approaches have high potential (performing strongly on the Zapier recipes) and can serve as a promising approach to more complex program synthesis challenges.

In this paper we explore the task of program synthesis from natural language. The goal is to translate a natural language description into an executable program. We specifically consider the use case of If-Then program synthesis as it aligns closely with enterprise automation. Business processes are driven by the collection, movement, and synthesis of information produced by various information systems. Given the technical complexity and programmatic interfaces of these information systems, enterprise automation is often bottlenecked by engineering and IT effort. Non-technical business users operate at the level of business logic and high level concepts. There can be significant value in developing intelligent systems to empower non-technical business users and provide them with the ability to create automations from natural language descriptions.

On the consumer side, services like Zapier and If Then Than That (IFTTT) allow non-technical users to easily chain together various web services and APIs to create automations. These automations are described as "recipes" and are generated by the end user using a graphical user interface. Recipes on IFTTT and Zapier follow the If-Then format. The user selects a triggering event on a web service and an ensuing action to be executed (using the same or different web service). For example, imagine a user who wishes to automatically publish breaking news on twitter. A simple

If-Then recipe to achieve this could be: if a new article is posted on NY Times, publish a tweet with link to article on my Twitter.

Both IFTTT and Zapier use a point and click graphical interface which are quite intuitive. We consider the use case of generating automation recipes from natural language descriptions for two reasons. With the proliferation of chatbots and dialog systems, business users can quickly and more efficiently describe an automation using natural language. Additionally, the compactness of languages allows for the description of more complex business processes which can often be more difficult to model visually.

In this paper, we aim to predict IFTTT and Zapier recipes from natural language descriptions. In contrast to previous approaches, we frame the task an end-to-end sequence-to-sequence learning problem. All associated code and data for this paper can be found on the author's github page: https://github.com/dhairyadalal/nmt_if_then

## 1 If-Then Program Synthesis

If-Then programs consist of a triggering event and an ensuing action. In this paper we explore trigger-action programs in the context of IFTTT and Zapier "recipes". We use the IFTTT (Quirk, Mooney, and Galley 2015) and Zapier (Chen et al. 2016) data sets. While the specific terminology for IFTTT and Zapier recipes are slightly different, the recipe formats are conceptually the same. The automation recipes consist of channel entities, functions, and function arguments. Channel entities are the specific web service providers such as Twitter or New York Times. Functions are the API services provided by the channel entities. Functions can also take in arguments. Following with previous work on the IFTTT, we did not consider argument extraction in this paper.

There are many interesting challenges that arise in If-Then program synthesis. Given the examples in Figure 1, we can see following challenges:

- Alignment: the trigger/event channel appears in different positions across the two examples. In the IFTTT example the "If" condition is mentioned first and in the Zapier

**IFTTT Example**

**Title:** Add *Checkin* to *Evernote*
**Description:** N/A

**Recipe:**
      **Foursquare:** New check-in with Photo **Evernote:** Create image note from URL
      {**Trigger Channel,** Trigger Function, **Action Channel,** Action Function}

**Sequence to Predict**
      *foursquare foursquare.new_check_in_with_photo evernote evernote.create_image_note_from_url*

**Zapier Example**

**Title:** *HipChat* Alert from *RSS* Feed
**Description:** Get a Hipchat message for new RSS feed items.

**Recipe:**
      **RSSAPI:** New Feed **HipChatAPI:** Message
      {**Event Channel,** Event, **Action Channel,** Action}

**Sequence to Predict**
      *rssapi rssapi.new_feed hipchatapi hipchatapi.message*

Figure 1: Example recipes from IFTTT and Zapier

example, it appears second.

- Channel/function ambiguity: in the IFTTT example, Foursquare is not explicitly mentioned in the title. We infer the act of "checkin" to be associated with Foursquare checkins.

- Variable input lengths: descriptions of recipes are not the same length

## 1.1 IFTTT

In the IFTTT domain, an automation recipe consists of the following four elements: Trigger Channel, Trigger Function, Action Channel and Action Function. The trigger is the "If" part of the If-Then program and identifies a condition or activity that will trigger the "Then" or action that will ensue.

We obtained the IFTTT dataset and relevant materials from the Microsoft Open Data Repository. Quirk, Mooney, and Galley do not provide the scraped IFTTT recipes. Instead they provide their web scraping tool, a list of recipe urls, and the Mechanical Turk annotations for the test set. After scraping and extraction process, we have a dataset that contains the following for each recipe: recipe title, recipe description, and the formal IFTTT recipe. The quality of the recipe descriptions was highly variable and unreliable. We followed the convention of previous work and used the recipe title as our natural language description.

Since 2015, many of the provided urls are no longer viable. Additionally, upon inspecting the dataset, we further removed examples where the title was not in English or the title contained fewer than three words. Additionally, we followed the convention set by previous work and further pruned our test set based on the Mechanical Turkers annotations. We only selected examples where three or more annotators agreed on all the extracted recipe parts. Table 1 described the size of our final dataset. In previous works the test set was slightly larger at 684 examples, however not all of those example were available during our scraping activity.

## 1.2 Zapier

Zapier recipes are conceptually identical to IFTTT recipes and consist of the following four elements: Event Channel, Event, Action Channel, and Action. Here the *Event* and *Action* are identical to *Triggers* and *Actions* in the IFTTT domain.

| Type | Original | Removed | % Removed | Final |
|------|----------|---------|-----------|-------|
| train | 77,495 | 37,523 | 48 of the recipe | 39,972 |
| validation | 5,161 | 2,615 | 50 | 2,546 |
| test | 4,294 | 3,740 | 87 | 554 |

Table 1: IFTTT data after cleaning bad urls and bad entries.

We used Zapier dataset that was proposed in (Chen et al. 2016). The authors provided a segmented train and test dataset on their github repository. The Zapier has 17,264 train examples and 4,859 test examples. We further segment the train set by randomly generating a validation set of 3,896 examples. Like the IFTTT, each example consisted of a recipe title, recipe description, and the full recipe. Chen et al. do not provide performance metrics of their model on the Zapier dataset, but they do provide their code. We were unable to get baseline metrics of the Latent Attention model on the Zapier data.

## 2 Related Work

Program synthesis from natural language is a broad research space with numerous use cases and challenges. The IFTTT recipe prediction task was first proposed by Quirk, Mooney, and Galley. Quirk, Mooney, and Galley initially framed the problem as a structured prediction task where channels and thier corresponding functions were predicted separately using phrasal machine translation and text-similarity retrieval methods. Beltagy and Quirk 2016 then framed the problem as an ensemble based sequence-to-sequence prediction task. In this approach, distinct recurrent neural networks and a logistic regression classifiers were trained to predict the channels and functions as separate multi-class labels. The prediction were combined to form recipe candidates and an ensemble model was trained to select the best recipe candidate. Chen et al. 2016 proposed an end-to-end neural architecture. While channels and functions were treated as distinct multi-class labels, the model jointly predicted all the recipe components using shared latent attention. (Chen et al. 2016) also produced the Zapier dataset but did not publish any evaluation metrics as their paper focused primarily of the IFTTT use case. Finally, Yao et al. 2018 proposed an interactive reinforcement learning agent that asks a human user for clarification prior parsing the natural language description.

Outside of the If-Then program domain, various strategies and use cases have emerged. Lin 2017 first proposed a seq2seq approach for program synthesis using a RNN encoder-decoder to translate natural language commands into executable shell scripts. Krishnamurthy, Dasigi, and Gardner 2017 developed an augmented encoder-decoder model that incorporates type-checking and entity resolutions to improve question answering on semi-structured tables. In uses cases where there is an execution environment available, reinforcement learning and weakly supervised approaches have also been proposed.

## 3 Program Synthesis as Seq2Seq Learning

In contrast to previous approaches, we consider framing If-Then program synthesis as an end-to-end sequence learning task. The sequence-to-sequence encoder-decoder architecture was first introduced by Sutskever, Vinyals, and Le 2014 for machine translation. Given a sequence of inputs $\{x_1, .., x_i\}$ where $x \in Vocabulary_{source}$, 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 (Kreutzer, Bastings, and Riezler

|  | 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.

2019). The encoder and decoder are bidirectional LSTMs with multiplicative attention (Luong, Pham, and Manning 2015). 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.

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 Open-NMT model provided in OpenNMT-py library (Klein et al. 2018). 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 Klein et al. for further model specifics.

### 4.3 Basic Transformer

The final model we evaluated was the Transformer model introduced in Vaswani et al. 2017. We used the JoeyNMT framework to implement and train the Transformer model. The model consisted of six layer transfor of the architecture-mer encoder and decoder with eight attention heads and used additive attention (Bahdanau, Cho, and Bengio 2014). 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 (Shazeer and Stern 2018) 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

| | 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.

| | 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.

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 Chen et al. 2016 and Beltagy and Quirk 2016. 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, Open-NMT 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 Chen et al. 2016 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 about the intended channel functions.

# 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.

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 metrics. It had an sequence accuracy score of 93.91%, positional accuracy of 97.4%. In the future, we plan expand the scope our task to account for argument extraction and explore more complex program synthesis challenges.

# References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation.

Beltagy, I., and Quirk, C. 2016. Improved semantic parsers for if-then statements. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 726–736. Berlin, Germany: Association for Computational Linguistics.

Chen, X.; Liu, C.; Shin, R.; Song, D.; and Chen, M. 2016. Latent attention for if-then program synthesis. In *Proceedings of the 29th Advances in Neural Information Processing Systems*.

Klein, G.; Kim, Y.; Deng, Y.; Nguyen, V.; Senellart, J.; and Rush, A. 2018. OpenNMT: Neural machine translation toolkit. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, 177–184. Boston, MA: Association for Machine Translation in the Americas.

Kreutzer, J.; Bastings, J.; and Riezler, S. 2019. Joey NMT: A minimalist NMT toolkit for novices. *To Appear in EMNLP-IJCNLP 2019: System Demonstrations*.

Krishnamurthy, J.; Dasigi, P.; and Gardner, M. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 1516–1526. Copenhagen, Denmark: Association for Computational Linguistics.

Lin, X. V. 2017. Program synthesis from natural language using recurrent neural networks.

Luong, M.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. *CoRR* abs/1508.04025.

Quirk, C.; Mooney, R.; and Galley, M. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 878–888. Beijing, China: Association for Computational Linguistics.

Shazeer, N., and Stern, M. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *CoRR* abs/1804.04235.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. *CoRR* abs/1409.3215.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 5998–6008.

Yao, Z.; Li, X.; Gao, J.; Sadler, B. M.; and Sun, H. 2018. Interactive semantic parsing for if-then recipes via hierarchical reinforcement learning. *CoRR* abs/1808.06740.