# Text2App: A Framework for Creating Android Apps from Text Descriptions

**Masum Hasan[1], Kazi Sajeed Mehrab[2], Wasi Uddin Ahmad[3], Rifat Shahriyar[2]**

[1]Department of Computer Science, University of Rochester, [2]Bangladesh University of Engineering and Technology, [3]University of California, LA

## INTRODUCTION

We present Text2App -- a framework that allows users to create functional Android applications from natural language specifications.

**Natural language description:**
Create an app with a textbox, a button named "Speak", and a text2speech. When the button is clicked, speak the text in the text box.

**Simplified App Representation:**

```
<complist>
  <textbox>
  <button> STRING0
</button>
  <text2speech>
</complist>

<code>
  <button1_clicked>
    <speak>
      <textbox1text>
    </speak>
  </button1_clicked>
</code>
```

**Literal Dictionary:**

```
{
    "STRING0": "Speak"
}
```


Figure 1: An example app created by our system that speaks the textbox text on button press.

You can watch a demo, or try out our software!
https://text2app.github.io/

SCAN ME

## The Problem

**Deep Learning models suffers from long input. But softwares are very long!**

Popular Natural Language Processing models, such as Transformer [4], has memory requirements are quadratic in respect to the size of their inputs.

There have been many attempts at program synthesis from Natural Language recently, however, they are limited to generating a single method.

## The Idea

Large softwares are often modular, and large chunk of it perform an uniform functionality. We can group parts of code into single tokens and fit into the Deep Learning model context window.

We build a ***custom formal language***, and ***a compiler***, that helps our seq2seq neural network model to predict large software.

## METHOD

### Simplified App Representation (SAR)

We developed a formal language named Simplified App Representation or SAR, that summarized an app functionality in a smaller number of tokens. The Context Free Grammar for the language is given below. An example SAR at Fig. 1.

$\langle SAR \rangle \rightarrow \langle screens \rangle$
$\langle screens \rangle \rightarrow \langle screen \rangle$ '<NEXT>' $\langle screens \rangle \mid \langle screen \rangle$
$\langle screen \rangle \rightarrow \langle complist \rangle \langle code \rangle$
$\langle complist \rangle \rightarrow$ '<COMPLIST>' $\langle comps \rangle$ '</COMPLIST>'
$\langle comps \rangle \rightarrow \langle comps \rangle \langle comp \rangle \mid \langle comp \rangle$
$\langle comp \rangle \rightarrow$ '<COMP>' $\langle args \rangle$ '</COMP>' | '<COMP>'
$\langle code \rangle \rightarrow$ '<CODE>' $\langle events \rangle$ '</CODE>'
$\langle events \rangle \rightarrow \langle event \rangle \langle events \rangle \mid \langle event \rangle$
$\langle event \rangle \rightarrow$ '<EVENT>' $\langle actions \rangle$ '</EVENT>'
$\langle actions \rangle \rightarrow \langle actions \rangle \langle action \rangle \mid \langle action \rangle$
$\langle action \rangle \rightarrow$ '<ACTION>' $\langle args \rangle$ '</ACTION>'
$\langle args \rangle \rightarrow \langle arg \rangle \langle args \rangle \mid \langle arg \rangle$
$\langle arg \rangle \rightarrow$ '<ARG>' '<VAL>' '<ARG>' | '<VAL>'

Fig. 1. CFG for SAR

### SAR Compiler

We created a custom compiler that can turn a SAR representation of an app to an app source code. We used MIT App Inventor 2 (AI2) as a backend for compiling the source code into an actual functional Android application.

### Data Synthesizer

Collecting data for this task is cumbersome. Hence, we performed a survey among 30 participants to identify how people like to provide natural language inputs. Then we designed a data synthesizer that automatically generates Natural Language descriptions of apps and their valid SAR representation that can be compiled into app. The data synthesizer works as below.

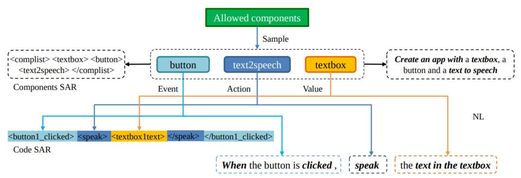



Figure 2. Automatic synthesis of NL and SAR parallel corpus. Bold-italic indicates text is selected stochastically.

### Augmenting Synthetic Data with BERT

Synthetic data often has a homogenous distribution, and may not contain the same distribution of natural language from the wild. To alleviate the problem, we created a BERT Mutation method that randomly mutates 10% tokens with a contextually relevant token sampled from the probability distribution of BERT masked language modeling. The following table shows an example of BERT Mutation based data augmentation.

| |
|---|
| **Original:** Create an app that has an audio player with source string0, a switch. If the switch is flipped, play player. |
| **Augmentation:** Create an app that has an external player with source string0, a switch. If the switch gets flipped, play player. |

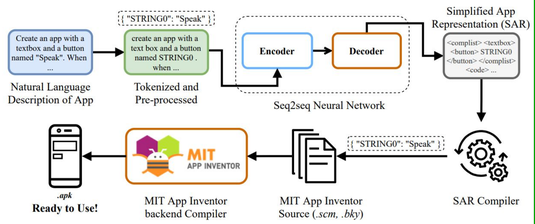
Table 1. BERT mask filling based data augmentation method. Mutated words are highlighted green.

### Training a Seq2Seq Transformer model

We generate 50,000 unique NL and SAR parallel data using our data synthesis method, and mutate 1% of the natural language tokens. We split this dataset into train-validation-test sets in 8:1:1 ratio and train three different models: Pointer Network [1], seq2seq Transformer models with encoder weight from CodeBERT [2], and RoBERTa [3].

## Full Pipeline:

Once trained, our entire app development pipeline looks like this,




Given any Natural Language input, our seq2seq model translates it into a SAR, which is a simpler description of the app. Furthermore, the SAR compiler and MIT App Inventor 2, compiles the SAR into functional apk.

### Creating app from Abstract language

In our survey, human evaluators often describe app that the model cannot understand without having external context. Such as, *"Make google front page"*. We experiment with GPT-3 [5] to solve such cases.

## Results

Our seq2seq model performs fairly well when tested on a held out synthetic test set.

| | #Epoch | Test | | BERT Mutation 2% | | BERT Mutation 5% | | BERT Mutation 10% | | Unseen Pair | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | BLEU | EM | BLEU | EM | BLEU | EM | BLEU | EM | BLEU | EM |
| **Without Training Data Augmentation** | | | | | | | | | | | |
| **PointerNet** | 13.6 | 94.64 | 79.24 | 94.16 | 72.06 | 91.80 | 56.14 | 88.96 | 40.78 | 96.75 | 82.91 |
| **RoBERTa init** | 3 | 97.20 | 77.80 | 96.83 | 73.20 | 94.97 | 61.68 | 92.86 | 48.06 | 98.11 | 79.66 |
| **CodeBERT init** | 8 | 97.42 | 80.02 | 97.18 | 76.02 | 95.37 | 64.38 | 93.29 | 51.24 | 98.47 | 83.50 |
| **With Training Data Augmentation (1% Mutation)** | | | | | | | | | | | |
| **PointerNet** | 23.2 | 95.03 | 81.40 | 94.85 | 79.46 | 93.85 | 72.04 | 92.53 | 63.68 | 96.68 | 83.33 |
| **RoBERTa init** | 3 | **97.66** | **81.76** | **97.60** | **80.66** | **96.91** | **76.16** | **96.04** | **70.10** | **98.64** | **84.68** |
| **CodeBERT init** | 7 | 97.64 | 81.66 | 97.51 | 80.20 | 96.74 | 74.98 | 95.71 | 67.58 | 98.62 | 84.51 |

Table 2. Seq2seq model performances.

### Human Evaluation

To evaluate our system, with real world natural language, we conduct a, survey with 13 Computer Science undergraduate, volunteers. We found a BLEU-1 score of 54.99, and exact match 4/111. Table 3 shows our observation from the human evaluation

| Observation | Example |
|---|---|
| Successful predictions closely follow our data format. (i.e. clear component list followed by functionality.) | **NL:** *Create an app that has a button named "Take Photo", when clicked open the rear camera and capture an image.* |
| Many predictions are correct, but the human label is wrong. | **NL:** *Create an app that can take a photo with the camera.* (Missing mention of a button.) |
| User expects the automatic system to have inherent understanding of the real world. | **NL:** *A typical registration form with necessary text fields and a submit button.* |
| Model is biased towards synthesizer keywords. | **NL:** *Create an app which has a moving ball.* ('moving' frequently appears with accelerometer, model predicts accelerometer component.) |
| System misses components not specified. | **NL:** *Create an app that has a textbox labelled "Insert Text", when the device is shaken, speak the text in the textbox* |

Table 3. Observation from human evaluation

## Limitation and Discussion

Our method shows promising results for building a "Natural Language Compiler" that allows user to develop app with natural language text instead of a programming language. More work is needed to make neural network make formal predictions, and adding more functionalities to the compiler.

## References

[1] Abigail See, Peter J. Liu, and Christopher D. Manning., 2017. Get to the point: Summarization with pointergenerator networks. In Proceedings of the 55th Annual Meeting of the Association for Computational, Linguistics (Volume 1: Long Papers), pages 1073–, 1083, Vancouver, Canada. Association for Computational Linguistics.

[2] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin,, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A pre-trained model for programming and, natural languages. In Findings of the Association, for Computational Linguistics: EMNLP 2020, pages 1536–1547, Online. Association for Computational
Linguistics.

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis,, Luke Zettlemoyer, and Veselin Stoyanov. 2020., Ro{bert}a: A robustly optimized {bert} pretraining, approach.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob, Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz, Kaiser, and Illia Polosukhin. 2017. Attention is all, you need. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie, Subbiah, Jared D Kaplan, Prafulla Dhariwal,, Arvind Neelakantan, Pranav Shyam, Girish Sastry,, Amanda Askell, Sandhini Agarwal, Ariel HerbertVoss, Gretchen Krueger, Tom Henighan, Rewon, Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu,, Clemens Winter, Chris 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. In, Advances in Neural Information Processing Systems,, volume 33, pages 1877–1901. Curran Associates,, Inc.

UNIVERSITY of ROCHESTER