

Intern Final Presentation

Haocheng Han
09/27/2019



Overview of Internship



Email Bot



Lggen



Grammar Checker




Email Bot

Aim:

- Archive an Email Bot based on declarative methods
- Test the expressions and built-in functions

Features:

- Show Emails in Inbox/Sent Items
 - Get Email From Graph
 - Show Emails in pages/Next/Last Pages
 - Select Emails: Reply/Forward/Go back
 - Reply Emails
 - Delete data
 - Reply through Graph
 - Forward Emails
 - Delete data
 - Reply through Graph
 - Find Contacts
 - Delete data
 - find contact through Graph
- 



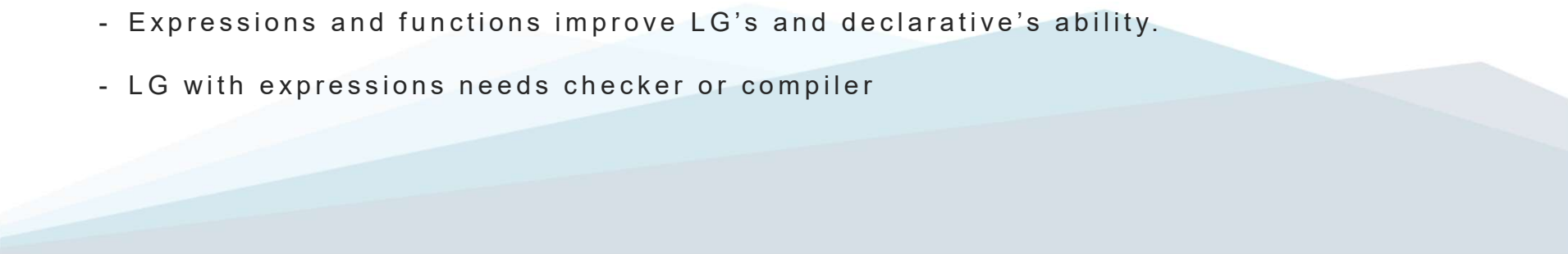
Email Bot

```
# ShowRecipient(user)
- IF: {count(user.intentRecipient.value) == 1}
  - Cool, I Find the Recipient You Want, it's {user.intentRecipient.value[0].displayName}.
- ELSEIF: {less(count(user.intentRecipient.value), 0)}
  - Emm, I didn't Find What you Need, Can You Try it Again ?
- ELSEIF: {greater(count(user.intentRecipient.value), 1)}
  - I Find Several People Satisfy Your Demand. The First Two People are {user.intentRecipient.value[0].displayName} and {user.intentRecipient.value[1].displayName}.
- ELSE:
  - Something Strange Happened.
```

Expressions:

- String functions: length, toLower/Upper, addOrdinal, concat, split
- Collection functions: contains, count, join, foreach
- Logical comparison functions: and, or, not, equals, less,
- Conversion functions: int, string, bool, binary
- Math functions: add, div, rand, max, range
- Date and time functions: formatDateTime, utcNow, convertFromUTC
- URI parsing functions: uriHost, uriPath, uriPort, uriQuery
- Object manipulation and construction functions: setProperty

Experience:

- Expressions and functions improve LG's and declarative's ability.
 - LG with expressions needs checker or compiler
- 

Email Bot

Migrating to composer:

- extract all LG variables into one file
- all files are flat
- how to show Bot's structure in composer?

Experience and feeling:

- Easy to learn and use, like programming language
- LG helps to separate Bot's logic and output
- The separation sometimes depends on concrete situation

| | |
|------------------------|---------------|
| function | findmycontact |
| graph | forward |
| EmailDialog.lg | reply |
| EmailPage.dialog | send |
| Expression.docx | showinbox |
| Initialization.dialog | showsentitems |
| MyEmailBot.main.dialog | |

| |
|----------------------------|
| common |
| DeleteContactInfo |
| DeleteForwardInfo |
| DeleteReplyInfo |
| DeleteSendInfo |
| EmailPage |
| FindMyContact |
| ForwardMyEmail |
| ForwardMyEmailThroughGraph |
| GetMyEmail |
| GetMyEmailFromGraph |
| GetMySentItems |
| GetRecipientFromGraph |
| GetSentItemsFromGraph |
| Initialization |
| Main |
| ReplyMyEmail |
| ReplyMyEmailThroughGraph |
| SendMyEmail |
| SendMyEmailToGraph |
| ShowMyCurrentEmail |
| ShowMyEmail |
| ShowMyLastPage |
| ShowMyNextPage |
| ShowSentItems |
| ShowSentLastPage |
| ShowSentNextPage |
| SignInToGraph |



LGgen

Solving Problem:

- To call a LG template, we need to spell its full name.
- Incorrect/missing template name will lead to a runtime error, hard to find.

What it is:

A LG CLI to generate a class file from a LG file or a folder, supporting C# and Typescript.

Aim:

- Transfer the runtime error into compile error
- Use VS IntelliSense to input the template name automatically
- User can refer to the LG template more easily and efficiently, also reduce error in LG usage.

Features:

- Generate C#/TS class file from a file or a folder
- Support grammar check mode

[Demo](#)



LGgen

Improve structure and refactoring:

- Decouple language support with an interface
- Decouple command handler with an abstract class
- Use simple factory mode to manage modules, 'register' for languages and 'builder' for commands

```
4 references
public interface ILanguageGenerator
{
    3 references
    void Generate(string outputPath, string className, List<LGTemplate> temp);
}
```

```
6 references
public BaseCommand(string usage, string usageSample, string command, bool singleCommand = true)
{
    Usage = usage;
    UsageSample = usageSample;
    Command = command;
    SingleCommand = singleCommand;
}
```

```
1 reference
public static void RegisterAllLanguages()
{
    Register("cs", ".cs", typeof(CSharpGenerator));
    Register("ts", ".ts", typeof(TypescriptGenerator));
}
```

```
public List<BaseCommand> Commands { get; set; }

1 reference
public CommandHandler()
{
    Messages = new List<string>();
    LGFiles = new List<string>();
    Commands = new List<BaseCommand>();
    Language = "cs"; // default cs
    LanguageRegister.RegisterAllLanguages();
}

6 references
public CommandHandler AddCommand (BaseCommand command)
{
    Commands.Add(command);
    return this;
}
```

Experience:

- Code should be consist of decoupled modules inherited from interfaces and abstract classes, complying to the Open Closed Principle: open for extension, close for modification.
- Naming Guideline. The name of variables, methods and class should represent their intension and usage.

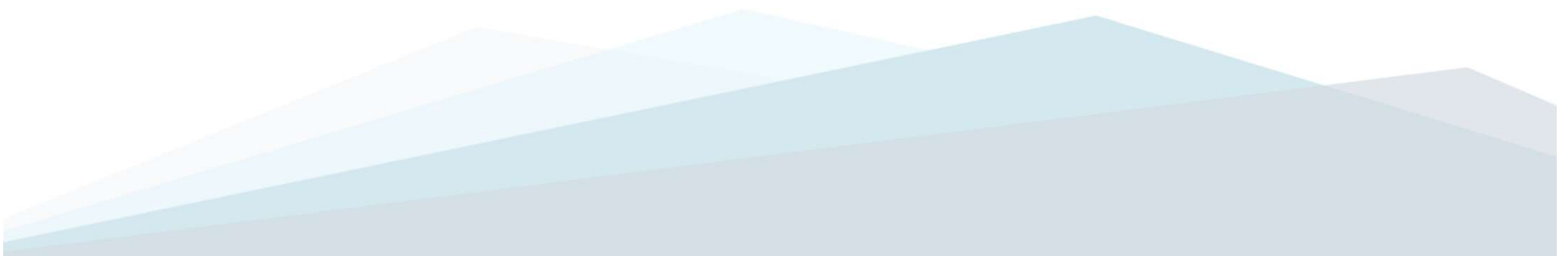


Grammar Checker

Grammar Checker is to find basic syntax error in sentences, like the check function in Word, or Grammarly.

The traditional way to solve this problem is to analysis the syntax component of the sentence. For example, Link Grammar of CMU is one of the products.

Also, SyntaxNet from Google uses AI-based method to solve this problem.



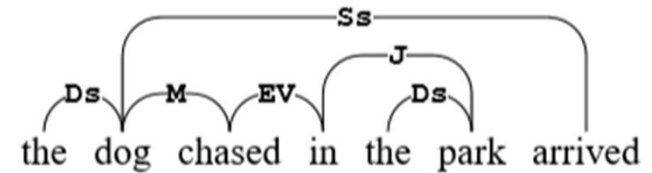
Grammar Checker

Link Grammar:

Using links to find if a sentence is valid.

If all links in a sentence satisfy following rules, this sentence is valid:

1. Planarity. Different links should not cross.
2. Connectivity. Links should link all words in a sentence.
3. Exclusion. No two link should link the same pair of words.
4. Ordering. When the connector in the formula are from left to right, the word they link to are from near to far.



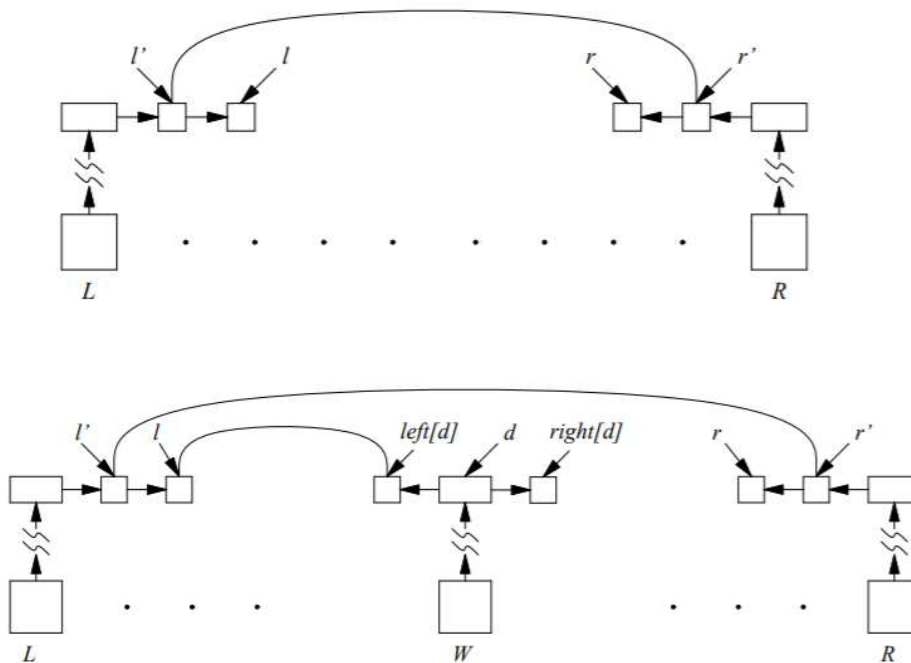
a word's formula `dog: { @A+ } & Ds- & { @M- } & (Ss+ or SIs- or 0- or J-);`

a disjunct `((L1, L2, ..., Lm) (Rn, Rn-1, ..., R1))` like `d=((D,0) ())`

a connector `A/L1/R1`

Grammar Checker

Parse the sentence: Count all possible links in the whole sentence. If it > 0 , this sentence is valid.



```

PARSE
1   $t \leftarrow 0$ 
2  for each disjunct  $d$  of word 0
3    do if  $left[d] = \text{NIL}$ 
4      then  $t \leftarrow t + \text{COUNT}(0, N, right[d], \text{NIL})$ 
5  return  $t$ 

COUNT( $L, R, l, r$ )
1  if  $L = R + 1$ 
2    then if  $l = \text{NIL}$  and  $r = \text{NIL}$ 
3      then return 1
4    else return 0
5  else  $total \leftarrow 0$ 
6    for  $W \leftarrow L + 1$  to  $R - 1$ 
7      do for each disjunct  $d$  of word  $W$ 
8        do if  $l \neq \text{NIL}$  and  $left[d] \neq \text{NIL}$  and  $\text{MATCH}(l, left[d])$ 
9          then  $leftcount \leftarrow \text{COUNT}(L, W, next[l], next[left[d]])$ 
10         else  $leftcount \leftarrow 0$ 
11        if  $right[d] \neq \text{NIL}$  and  $r \neq \text{NIL}$  and  $\text{MATCH}(right[d], r)$ 
12          then  $rightcount \leftarrow \text{COUNT}(W, R, next[right[d]], next[r])$ 
13         else  $rightcount \leftarrow 0$ 
14         $total \leftarrow total + leftcount * rightcount$ 
15      if  $leftcount > 0$ 
16        then  $total \leftarrow total + leftcount * \text{COUNT}(W, R, right[d], r)$ 
17      if  $rightcount > 0$  and  $l = \text{NIL}$ 
18        then  $total \leftarrow total + rightcount * \text{COUNT}(L, W, l, left[d])$ 
19  return  $total$ 
    
```

The image features abstract geometric shapes in various shades of blue. In the top-left corner, there is a series of overlapping triangles pointing towards the top-right. At the bottom, there is a range of overlapping mountain-like shapes, each with a different blue hue, creating a layered effect.

THANKS FOR
LISTENING