

# **Разработка твиттер ботнета на основе цепей Маркова**

**ИУ5И-21М Ян Тяньци**

Перевод “Developing a Twitter botnet based on Markov chains in Go”

Основная идея этой статьи - рассказать как написать твиттер ботнет с автономными ботами которые смогут отвечать на другие твиты текстом сгенерированным с помощью алгоритма цепей Маркова. Так как это обучающий минипроект, то мы будем делать все сами и с самого нуля.

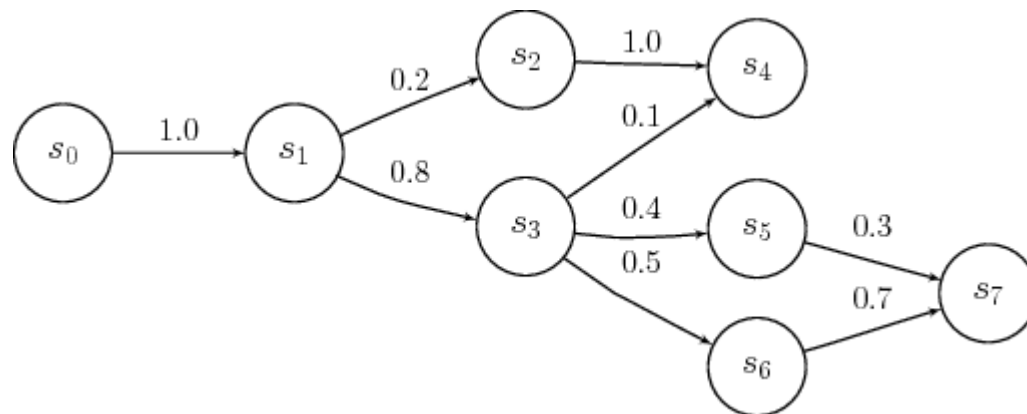
Идея совместить алгоритм цепей Маркова и твиттер ботов появилась после общения с x0rz.

## **Цепи Маркова**

Цепь маркова это последовательность стохастических событий(основанных на вероятности) где текущее состояние переменной или системы не зависит только от предыдущего события и не зависит от всех остальных прошедших событий.

[https://en.wikipedia.org/wiki/Markov\\_chain](https://en.wikipedia.org/wiki/Markov_chain)

В нашем случае мы будем использовать цепочки маркова для анализа вероятности что после некоторого слова идет другое определенное слово. Нам нужно будет сгенерировать граф, вроде того что на рисунке ниже, только с тысячами слов.



На вход нам нужно подавать документ с тысячами слов для более качественного результата. Для нашего примера мы будем использовать книгу Иммануила Канта “The Critique of Pure Reason”. Просто потому что это первая книга которая мне попалась в текстовом формате.

## Расчет цепи

Прежде всего нам нужно прочитать файл

```
copy1func readTxt(path string) (string, error) {
2    data, err := ioutil.ReadFile(path)
3    if err != nil {
4        //выполняем необходимую работу
```

```

5     }
6     dataClean := strings.Replace(string(data), "\n", " ", -1)
7     content := string(dataClean)
8     return content, err
9}

```

Для вычисления вероятности состояний нам нужно написать функцию, которая будет на вход принимать текст, анализировать его и сохранять состояния Маркова.

```

copy 1func calcMarkovStates(words []string) []State {
2     var states []State
3     // считаем слова
4     for i := 0; i < len(words)-1; i++ {
5         var iState int
6         states, iState = addWordToStates(states, words[i])
7         if iState < len(words) {
8             states[iState].NextStates, _ =
addWordToStates(states[iState].NextStates, words[i+1])
9         }
10
11         printLoading(i, len(words))
12     }

```

```

13
14     // считаем вероятность
15     for i := 0; i < len(states); i++ {
16         states[i].Prob = (float64(states[i].Count) /
float64(len(words)) * 100)
17         for j := 0; j < len(states[i].NextStates); j++ {
18             states[i].NextStates[j].Prob =
(float64(states[i].NextStates[j].Count) / float64(len(words)) * 100)
19         }
20     }
21     fmt.Println("\ntotal words computed: " +
strconv.Itoa(len(words)))
22     return states
23 }

```

Функция `printLoading` выводит в терминал прогрессбар просто для удобства.

```

copy 1func printLoading(n int, total int) {
2     var bar []string
3     tantPerFourty := int((float64(n) / float64(total)) * 40)
4     tantPerCent := int((float64(n) / float64(total)) * 100)
5     for i := 0; i < tantPerFourty; i++ {

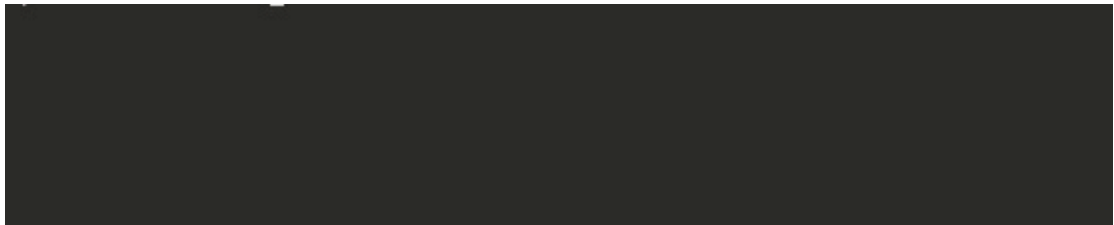
```

```

6         bar = append(bar, "■")
7     }
8     progressBar := strings.Join(bar, "")
9     fmt.Printf("r " + progressBar + " - " + strconv.Itoa(tantPerCent)
+ "")
10}

```

И выглядит это вот так:



## Генерация текста по цепи Маркова

Для генерации текста нам нужно первое слово и длина генерируемого текста. После этого запускается цикл в котором мы выбираем слова по вероятностям, рассчитанным при составлении цепи на прошлом шаге.

```

copy 1func (markov Markov) generateText(states []State, initWord string,
count int) string {
2     var generatedText []string
3     word := initWord
4     generatedText = append(generatedText, word)

```

```

5     for i := 0; i < count; i++ {
6         word = getNextMarkovState(states, word)
7         if word == "word no exist on the memory" {
8             return "word no exist on the memory"
9         }
10        generatedText = append(generatedText, word)
11    }
12    text := strings.Join(generatedText, " ")
13    return text
14}

```

Для генерации нам нужна функция, которая принимает на вход всю цепь и некоторое слово, а возвращает другое слово на основе вероятности:

```

copy 1func getNextMarkovState(states []State, word string) string {
2    iState := -1
3    for i := 0; i < len(states); i++ {
4        if states[i].Word == word {
5            iState = i
6        }
7    }
8    if iState < 0 {

```

```

9         return "word no exist on the memory"
10    }
11    var next State
12    next = states[iState].NextStates[0]
13    next.Prob = rand.Float64() * states[iState].Prob
14    for i := 0; i < len(states[iState].NextStates); i++ {
15        if (rand.Float64()*states[iState].NextStates[i].Prob) >
next.Prob && states[iState-1].Word != states[iState].NextStates[i].Word
{
16            next = states[iState].NextStates[i]
17        }
18    }
19    return next.Word
20}

```

## Твиттер АПИ

Для работы с АПИ твиттера будем использовать пакет go-twitter

Нам нужно настроить стриминг соединение - мы будем фильтровать твиты по определенным словам, которые есть в нашем исходном наборе:

```

copy 1func startStreaming(states []State, flock Flock, flockUser
    *twitter.Client, botScreenName string, keywords []string) {
2    // Convenience Demux demultiplexed stream messages
3    demux := twitter.NewSwitchDemux()
4    demux.Tweet = func(tweet *twitter.Tweet) {
5        if isRT(tweet) == false && isFromBot(flock, tweet) ==
false {
6            processTweet(states, flockUser, botScreenName,
keywords, tweet)
7        }
8    }
9    demux.DM = func(dm *twitter.DirectMessage) {
10        fmt.Println(dm.SenderID)
11    }
12    demux.Event = func(event *twitter.Event) {
13        fmt.Printf("%#v\n", event)
14    }
15
16    fmt.Println("Starting Stream...")
17    // фильтруем все что нам нужно
18    filterParams := &twitter.StreamFilterParams{
19        Track: keywords,

```



```

20         StallWarnings: twitter.Bool(true),
21     }
22     stream, err := flockUser.Streams.Filter(filterParams)
23     if err != nil {
24         log.Fatal(err)
25     }
26     // получаем сообщения пока стрим не будет остановлен
27     demux.HandleChan(stream.Messages)
28 }

```

Теперь когда нам будет попадаться твит с искомыми словами, то будет срабатывать функция `processTweet` в которой генерируется ответ с помощью алгоритма, описанного выше:

```

copy 1func processTweet(states []State, flockUser *twitter.Client,
botScreenName string, keywords []string, tweet *twitter.Tweet) {
2     c.Yellow("bot @" + botScreenName + " - New tweet detected:")
3     fmt.Println(tweet.Text)
4
5     tweetWords := strings.Split(tweet.Text, " ")
6     generatedText := "word no exist on the memory"
7     for i := 0; i < len(tweetWords) && generatedText == "word no
exist on the memory"; i++ {

```

```

8         fmt.Println(strconv.Itoa(i) + " - " + tweetWords[i])
9         generatedText = generateMarkovResponse(states,
tweetWords[i])
10    }
11    c.Yellow("bot @" + botScreenName + " posting response")
12    fmt.Println(tweet.ID)
13    replyTweet(flockUser, "@"+tweet.User.ScreenName+"
"+generatedText, tweet.ID)
14    waitTime(1)
15}

```

И постим твит с помощью `replyTweet`:

```

copy 1func replyTweet(client *twitter.Client, text string,
inReplyToStatusID int64) {
2    tweet, httpResp, err := client.Statuses.Update(text,
&twitter.StatusUpdateParams{
3        InReplyToStatusID: inReplyToStatusID,
4    })
5    if err != nil {
6        fmt.Println(err)
7    }
8    if httpResp.Status != "200 OK" {

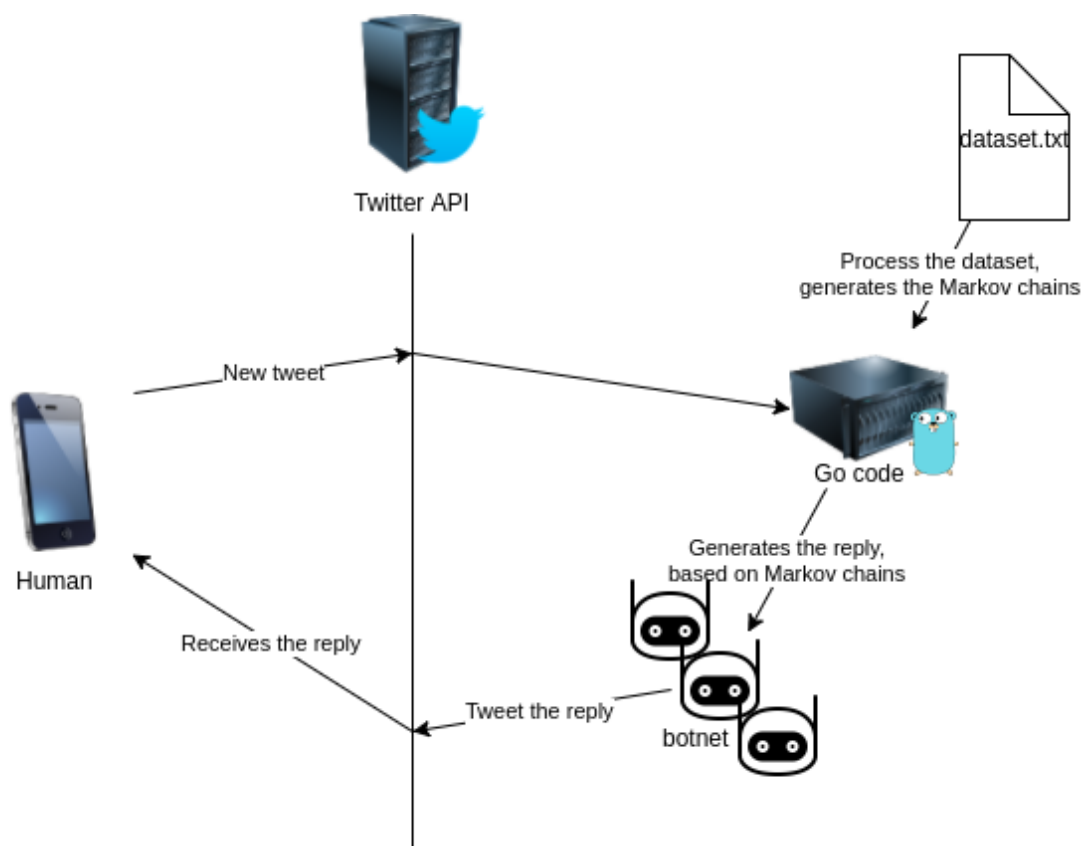
```

```
9         c.Red("error: " + httpResp.Status)
10         c.Purple("maybe twitter has blocked the account,
CTRL+C, wait 15 minutes and try again")
11     }
12     fmt.Print("tweet posted: ")
13     c.Green(tweet.Text)
14 }
```

### Стадный ботнет или как избежать ограничение твиттер АПИ

Если вы когда ни будь пользовались твиттер АПИ, то наверняка в курсе что есть целый ряд ограничений и лимитов. Это означает, что если ваш бот будет делать лишком много запросов, то его будут периодически блокировать на некоторое время.

Чтобы избежать этого мы будем использовать целую сеть ботов. Когда в стриме появится твит с нужным словом, один из ботов ответит на него и “уйдет в ждущий режим” на минуту, а обработкой следующих сообщений займутся другие боты. И так по кругу.



### Собираем все вместе

В нашем примере используются всего 3 бота. Это значит нам нужно три отдельных аккаунта. Ключи для этих аккаунтов вынесем в отдельный JSON файл который будем использовать как конфиг для нашего приложения.

```
сору 1[
2    {
3        "title": "bot1",
4        "consumer_key": "xxxxxxxxxxxxxx",
5        "consumer_secret": "xxxxxxxxxxxxxx",
```

```
6      "access_token_key": "xxxxxxxxxxxxx",
7      "access_token_secret": "xxxxxxxxxxxxx"
8  },
9  {
10     "title": "bot2",
11     "consumer_key": "xxxxxxxxxxxxx",
12     "consumer_secret": "xxxxxxxxxxxxx",
13     "access_token_key": "xxxxxxxxxxxxx",
14     "access_token_secret": "xxxxxxxxxxxxx"
15  },
16  {
17     "title": "bot3",
18     "consumer_key": "xxxxxxxxxxxxx",
19     "consumer_secret": "xxxxxxxxxxxxx",
20     "access_token_key": "xxxxxxxxxxxxx",
21     "access_token_secret": "xxxxxxxxxxxxx"
22  }
23]
```

## Демо

Мы настроили небольшую версию нашего ботнета с тремя ботами. Как уже говорилось, в качестве входных данных для

генерации цепи Маркова мы использовали книгу “The Critique of Pure Reason”.

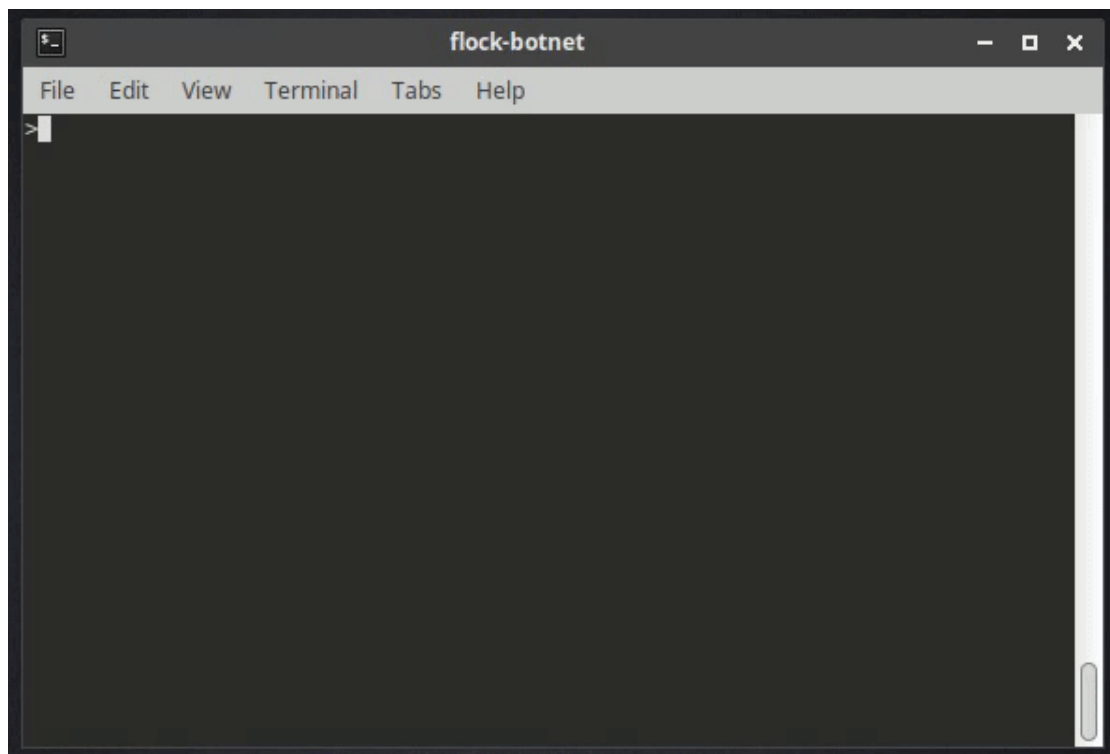
Когда ботнет запускается то все боты подключаются к стримингу и ждут когда появятся твиты с необходимыми ключевыми словами.

Каждый бот получает один из твитов, обрабатывает его и отправляет ответ с использованием цепи Маркова.

В терминале это выглядит вот так:

```
entry words to stream tweets (separated by comma):
human, reason, world, cognitive
total keywords: 4
keywords to follow: [human reason world cognitive]
Are you sure? [y/n]
y
ok, you are sure
Starting Stream...
waiting 35 seconds
2017-12-28 01:25:03.235255128 +0100 CET
bot @projectNSA - New tweet detected:
@AndyHerren I'll see Call Me, but doubt it will be as impactful for me. Looking
forward to 2 that aren't on your li... https://t.co/7XAD315nRS
0 - @AndyHerren
1 - I'll
2 - see
bot @projectNSA posting response
946175126969741312
tweet posted: @petals1031 see destroyed the mind. Human reason, in one sphere
of its cognition, is called
waiting 1 min to avoid twitter api limitation
2017-12-28 01:25:04.87310167 +0100 CET
```

И вот так выглядит все в процессе:



Ниже примеры твиттов, сгенерированные нашей цепью Маркова



**Jud** @Jud089 · 3m

Now she's against the not vampires too. After an easily seeable double cross over the kid she cares about for some reason?



1



**panopticon**

@projectNSA

Replying to @Jud089

Now logic presents me with principles, which it cannot decline, as they are presented by its



**Tim McAdams** @TimMcAdams1984 · 19m

My instincts tell me you're just another snowflake looking for ANY reason to diss on America. 🤔😂😏😏



1



**David Greco** @DaveGreco33 · 17m

Mine tell me, you support the #IsraelFirst traitors in my government & don't care how many US soldiers die in service to Israel #SyriaHoax



2



1




**panopticon**

@projectNSA

Replying to @DaveGreco33

tell him, when she was the mind. Human reason, in one sphere of its



 **Enzila** @enzilag1 · 34m  
Want to get an A? Check-out my study materials for Understanding Human Communication [studysoup.com/guide/2434107/...](https://studysoup.com/guide/2434107/...) t

← 2 ↻ ❤

 **andreimarkov**  
@andreimarkov00

Replying to @enzilag1

to consider questions, which it cannot decline, as they are presented by its cognition, is called

10:23 AM - 23 Apr 2017

← ↻ 📷 .ll

 **Camila Mendes** @maddimoselle · 1h  
I think Donald Duck is better because he can quack you out and you'll say, "I'd love to"

← 1 ↻ ❤ 1

 **andrew garfield** @andrwgrfld · 57m  
oh my god you'd better hide. Usually, Dickta becomes so aggressive when there's someone telling a punny joke.

← 1 ↻ ❤

 **Camila Mendes** @maddimoselle · 49m  
she has a new nickname now? I bet she wouldn't mind with my joke because I didn't talk about her chest.

← 1 ↻ ❤

 **dodecahedron**  
@dodecahedron00

 Follow

Replying to @maddimoselle

she was the mind. Human reason, in the mind. Human reason, in one sphere

## Заклучение

У нас получилось создать небольшой ботнет на основе алгоритма цепи Маркова, который может генерировать ответы на ТВИТЫ.

Мы использовали только 1 класс цепей маркова и сгенерированный текст не очень поход на настоящий человеческий. Но этого вполне достаточно для начала и в будущем можно будет использовать различные классы цепей маркова и другие техники для генерации более человеческого текста.

Твиттер АПИ может использоваться для самых различных целей. Надеюсь в будущем я смогу написать на эту тему еще несколько статей, например про анализ нод или пользователей и хештегов.