

Ghi chú của một coder

Vũ Anh

Tháng 01 năm 2018

Mục lục

Mục lục	2
1 Lời nói đầu	3
I Lập trình	5
2 Giới thiệu	6
2.1 Các vấn đề lập trình	6
2.1.1 Introduction	6
2.1.2 Data Structure	7
2.1.3 OOP	7
2.1.4 Networking	7
2.1.5 Sample Project Ideas	7
2.2 How to ask a question	8
2.3 Các vấn đề lập trình	8
2.4 Các mô hình lập trình	8
2.5 Testing	10
2.6 Logging	12
2.7 Lập trình hàm	13
2.8 Lập trình song song	14
2.9 IDE	15
3 Python	16
3.1 Giới thiệu	16
3.2 Cài đặt	17
3.3 Cơ bản	17
3.4 Cú pháp cơ bản	17
3.5 Yield and Generators	20
3.6 Cấu trúc dữ liệu	24
3.7 Quản lý gói với Anaconda	25
3.8 Test với python	25
3.9 Xây dựng docs với readthedocs và sphinx	26
3.10 Pycharm Pycharm	29
3.11 Vì sao lại code python?	29
4 Java	30
5 PHP	31

<i>MỤC LỤC</i>	2
II Xác suất	33
6 Các hàm phân phối thông dụng	34
6.0.1 Biến rời rạc	34
III Khoa học máy tính	36
7 Hệ điều hành	37
8 Ubuntu	38
IV Khoa học dữ liệu	39
9 Học máy	40
10 Học sâu	43
10.1 Tài liệu Deep Learning	43
10.2 Các layer trong deep learning	43
10.2.1 Sparse Layers	43
10.2.2 Convolution Layers	43
11 Xử lý ngôn ngữ tự nhiên	45
12 Nhận dạng tiếng nói	46
13 Phân loại văn bản	50
14 Pytorch	51
V Linh tinh	53
15 Nghiên cứu	54
16 Nghề lập trình	56
17 Latex	57
18 Chào hàng	59
19 Phát triển phần mềm	60
20 Phương pháp làm việc	61
Tài liệu	63
Chỉ mục	64

Chương 1

Lời nói đầu

Đọc quyển Deep Learning quá xá hay luôn. Rồi lại đọc SLP 2. Thấy sao các thánh viết hay và chuẩn thể (đấy là lý do các thánh được gọi là ... các thánh chẳng =))

Tính đến thời điểm này đã được 2 năm 10 tháng rồi. Quay lại với latex. Thỏa mãn được điều kiện của mình là một tool offline. Mình thích xuất ra pdf (có gì đọc lại hoặc tra cứu cũng dễ).

Hi vọng gắn bó với thằng này được lâu.

Chào từ hồi magizbox.wordpress.com, cái này tồn tại được 77 ngày (hơn 2 tháng) (từ 01/11/2017 đến 17/01/2018)

Chào Khách,

Mình là Vũ Anh. Tính đến thời điểm viết bài này thì đã lập trình được 7 năm (lập trình từ hồi năm 2010). Mình thích viết lách, bằng chứng là đã thay đổi host 2 lần datayo.wordpress.com, magizbox.com. Thành tựu ổn nhất hiện tại chỉ có một project [underthesea](https://github.com/magizbox/underthesea), xếp loại tạm được.

Blog này chứa những ghi chép loạn cào cào của mình về mọi thứ. Đúng với phong cách "vô tổ chức" của mình. Chắc chắn nó sẽ không hữu ích lắm với bạn. Tuy nhiên, cảm ơn bạn đã ghé qua.

Nếu Khách quan tâm, thì mình chỉ post bài xầm vào thứ 7 thôi nhé. Những ngày còn lại chỉ post bài nghiêm túc thôi. (bài này quá xầm nhưng được post vào thứ 5 nhé)

Làm sao để thực hiện blog này Viết mark-down và latex hỗ trợ bởi wordpress Server cho phép lưu ảnh động giphy Vấn đề lưu trữ ảnh: sử dụng tính năng live upload của github.com

Bỏ cái này vì quá chậm. Không hỗ trợ tốt latex (công thức toán và reference). Mình vẫn thích một công cụ offline hơn.

Chào từ hồi magizbox.com, cái này tồn tại được 488 ngày (1 năm 4 tháng. wow) (từ 01/07/2016 đến 01/11/2017)

Hello World,
 My name is Vu Anh. I'm a developer working at a startup in Hanoi, Vietnam. Coding and writing is fun, so I make this site to share my gists about computer science, data science, and more. It helps me keep my hobby and save information in case I forget. I wish it will be useful for you too.
 PS: I always looking for collaboration. Feel free to contact me via email brother.rain.1024[at]gmail.com
 Magizbox Stories
 Oct 2, 2016: Wow. It's 524th day of my journey. I added some notes in README.md, index.html, changed structure of website. Today I feel like at begin day when I start writing at datayo.wordpress.com blog. In this pass, there are times when I want to make a professional website like tutorialpoints but it doesn't work that way. Because in my heart, I don't want it, I don't to make a professional website. I just love coding, writing and sharing my hobby with people around the world. So today I come back to starting point, I will keep my writing schedule, make some fun stuffs each week.
 In July 2016, I turn to use HTML and mkdocs, and opensource magizbox.
 In March 2015, I start writing blog with wordpress.

Bỏ cái này vì thời gian build quá lằng nhằng. Quản lý dependencies các kiểu rất lâu. Muốn có một cái gì đó giúp viết thật nhanh và đơn giản.

Chào từ hồi datayo.wordpress.com, cái này tồn tại được 489 ngày (1 năm 4 tháng) (từ 01/03/2015 đến 01/07/2016)

I'm a junior data scientist, working as a researcher in big data team at a company in Vietnam. I love listening music when I'm writing code because it's make me coding better. I love reading books before sleeping because it take me sleep easier and discover the world with data mining via beautiful language R.
 I write this blog because I want to share my hobbies with everybody. I hope you will enjoy it. Feel free to contact me via twitter @rain1024oremailbrother.rain.1024@gmail.com(Iwillanswerallemailsforsure)foranythingyouwantto
 In case you find one of my posts could be better, don't hesitate to drop me a line in comment. I'm very appreciated and I will try my best to make it better and better.

Bỏ cái này. Bỏ wordpress. Vì muốn một site interactive hơn.

Phần I

Lập trình

Chương 2

Giới thiệu

2.1 Các vấn đề lập trình

I will to do crazy and dummy things, I will rewrite article for basic languages
(which tutorialpoints do very goods)

Each language I will cover these concepts:

Table of content

code/

1. introduction
2. syntax
3. data structure
4. oop
5. networking
6. os
7. parallel
8. event based
9. error handling
10. logging
11. configuration
12. documentation
13. test
14. ui
15. web
16. database
17. ide
18. package manager
19. build tool
20. make module
21. production (docker)

2.1.1 Introduction

Installation (environment, IDE)

Hello world

Courses

Resources

Syntax

variables and expressions

conditional

loops and Iteration

functions

define, use

parameters

scope of variables

anonymous functions

callbacks

self-invoking functions, inner functions

functions that return functions, functions that redefined themselves

closures

naming convention

comment convention

2.1.2 Data Structure

Number

String

Collection

DateTime

Boolean

Object

2.1.3 OOP

Classes Objects

Inheritance

Encapsulation

Abstraction

Polymorphism

For OOP Example: see Python: OOP

2.1.4 Networking

REST (example with chat app sender, receiver, message)

2.1.5 Sample Project Ideas

Guess My Number Game

Create Analog Clock

Create Pong Game

Create flappy bird

2.2 How to ask a question

Focus on questions about an actual problem you have faced. Include details about what you have tried and exactly what you are trying to do.

Ask about...

Specific programming problems

Software algorithms

Coding techniques

Software development tools

Not all questions work well in our format. Avoid questions that are primarily opinion-based, or that are likely to generate discussion rather than answers.

Don't ask about...

Questions you haven't tried to find an answer for (show your work!)

Product or service recommendations or comparisons

Requests for lists of things, polls, opinions, discussions, etc.

Anything not directly related to writing computer programs

2.3 Các vấn đề lập trình

Generic

KISS (Keep It Simple Stupid)

YAGNI

Do The Simplest Thing That Could Possibly Work

Keep Things DRY

Code For The Maintainer

Avoid Premature Optimization

Inter-Module/Class

Minimise Coupling

Law of Demeter

Composition Over Inheritance

Orthogonality

Module/Class

Maximise Cohesion

Liskov Substitution Principle

Open/Closed Principle

Single Responsibility Principle

Hide Implementation Details

Curly's Law

Software Quality Laws

First Law of Software Quality

2.4 Các mô hình lập trình

Main paradigm approaches 1

1. Imperative

Description:

Computation as statements that directly change a program state (datafields)

Main Characteristics:

Direct assignments, common data structures, global variables

Critics: Edsger W. Dijkstra, Michael A. Jackson

Examples: Assembly, C, C++, Java, PHP, Python

2. Structured

Description:

A style of imperative programming with more logical program structure

Main Characteristics:

Structograms, indentation, either no, or limited use of, goto statements

Examples: C, C++, Java, Python

3. Procedural

Description:

Derived from structured programming, based on the concept of modular programming or the procedure call

Main Characteristics:

Local variables, sequence, selection, iteration, and modularization

Examples: C, C++, Lisp, PHP, Python

4. Functional

Description:

Treats computation as the evaluation of mathematical functions avoiding state and mutable data

Main Characteristics:

Lambda calculus, compositionality, formula, recursion, referential transparency, no side effects

Examples: Clojure, Coffeescript, Elixir, Erlang, F, Haskell, Lisp, Python, Scala, SequenceL, SML

5. Event-driven including time driven

Description:

Program flow is determined mainly by events, such as mouse clicks or interrupts including timer

Main Characteristics:

Main loop, event handlers, asynchronous processes

Examples: Javascript, ActionScript, Visual Basic

6. Object-oriented

Description:

Treats datafields as objects manipulated through pre-defined methods only

Main Characteristics:

Objects, methods, message passing, information hiding, data abstraction, encapsulation, polymorphism, inheritance, serialization-marshalling

Examples: Common Lisp, C++, C, Eiffel, Java, PHP, Python, Ruby, Scala

7. Declarative

Description:

Defines computation logic without defining its detailed control flow

Main Characteristics:

4GLs, spreadsheets, report program generators

Examples: SQL, regular expressions, CSS, Prolog

8. Automata-based programming

Description:

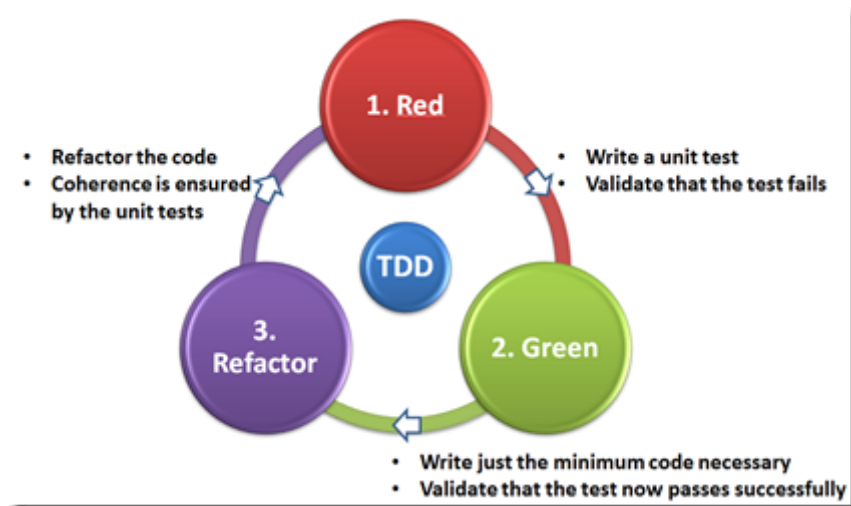
Treats programs as a model of a finite state machine or any other formal automata

Main Characteristics:

State enumeration, control variable, state changes, isomorphism, state transition table

Examples: AsmL

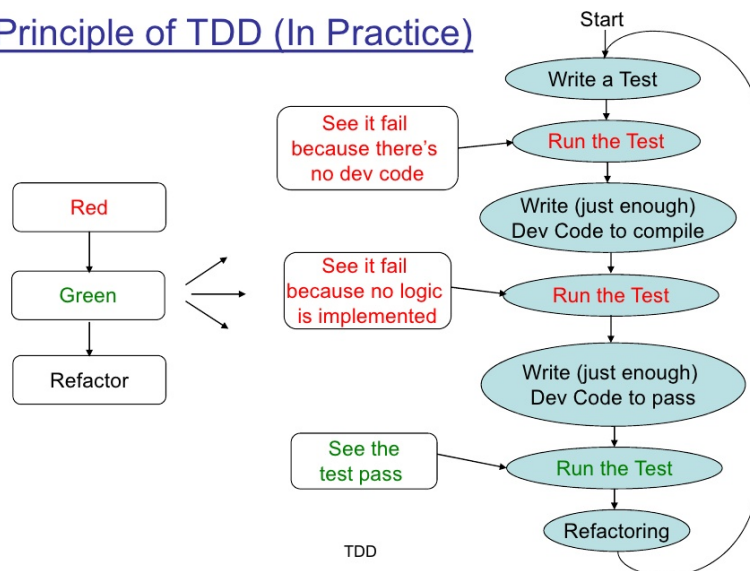
2.5 Testing



1. Definition 1 2

Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle:

Principle of TDD (In Practice)



Step 1: First the developer writes an (initially failing) automated test case

that defines a desired improvement or new function,

Step 2: Then produces the minimum amount of code to pass that test,

Step 3: Finally refactors the new code to acceptable standards.

Kent Beck, who is credited with having developed or 'rediscovered' the technique, stated in 2003 that TDD encourages simple designs and inspires confidence.

2. Principles 2

Kent Beck defines

Never with a single line of code unless you have a failing automated test.

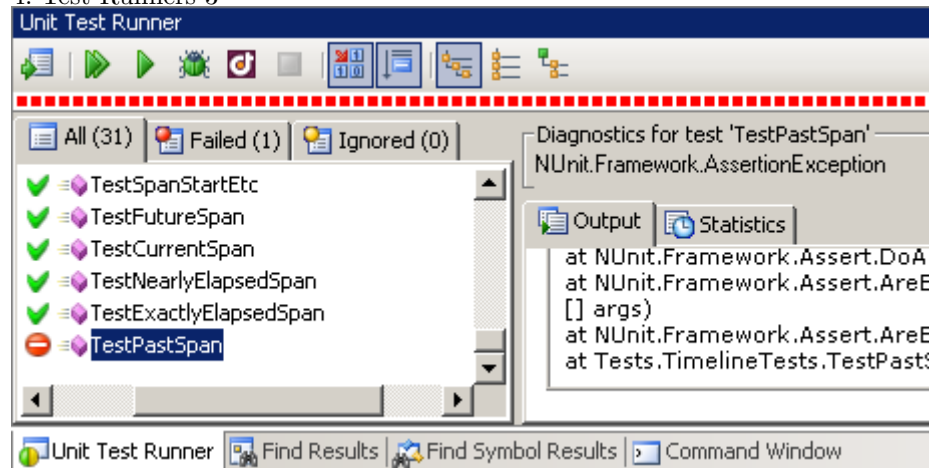
Eliminate duplication Red: (Automated test fail) Green (Automated test pass because dev code has been written) Refactor (Eliminate duplication, Clean the code)

3. Assertions Assert Framework

Numeric	Array	String	Exception
12, 34.5	[1, 2, 3] [4, 5, 6]	"hello" "world"	IOException TypeErrorException
areEqual	areEqual	areEqual	assertRaises
greaterThan	contains	startsWith	expected=Exception
lessThan	hasLength	endsWith	fail

Assert that the expected results have occurred. [code lang="java"] @Test
public void test() assertEquals(2, 1 + 1); [/code]

4. Test Runners 3



When testing a large real-world web app there may be tens or hundreds of test cases, and we certainly don't want to run each one manually. In such as scenario we need to use a test runner to find and execute the tests for us, and in this article we'll explore just that.

A test runner provides the a good basis for a real testing framework. A test runner is designed to run tests, tag tests with attributes (annotations), and provide reporting and other features.

In general you break your tests up into 3 standard sections; setUp(), tests,

and `tearDown()`, typical for a test runner setup.

The `setUp()` and `tearDown()` methods are run automatically for every test, and contain respectively:

The setup steps you need to take before running the test, such as unlocking the screen and killing open apps. The cooldown steps you need to run after the test, such as closing the Marionette session.

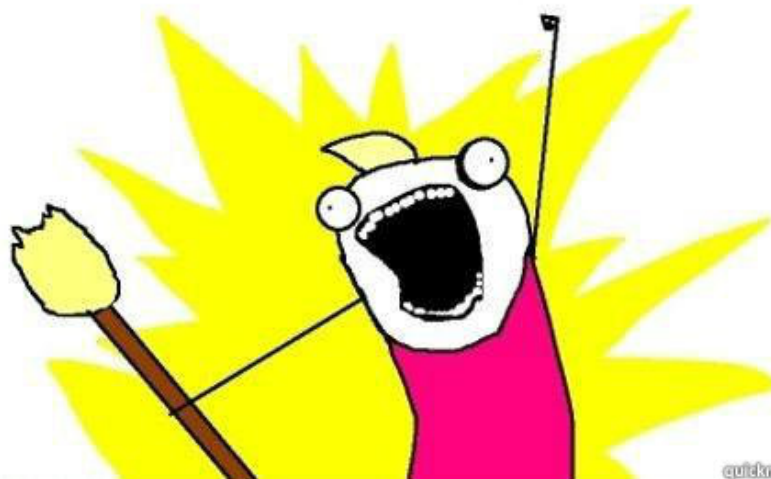
5. Test Frameworks

Language Test Frameworks C++/VisualStudio C++: Test Web Service rest-assured Web UI SeleniumHQ

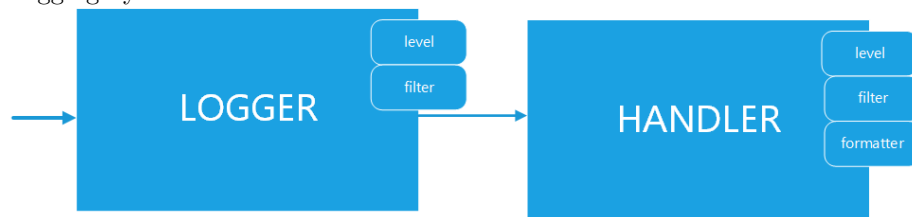
2.6 Logging

Logging is the process of recording application actions and state to a secondary interface.

LOG ALL THE THINGS



Logging System



Levels

Level When it's used DEBUG Detailed information, typically of interest only when diagnosing problems. INFO Confirmation that things are working as expected. WARNING An indication that something unexpected happened, or indicative of some problem in the near future (e.g. 'disk space low'). The software is still working as expected.

ERROR

Due to a more serious problem, the software has not been able to perform some function. **CRITICAL** A serious error, indicating that the program itself may be unable to continue running. Best Practices 2 4 5 Logging should always be considered when handling an exception but should never take the place of a real handler. Keep all logging code in your production code. Have an ability to enable more/less detailed logging in production, preferably per subsystem and without restarting your program. Make logs easy to parse by grep and by eye. Stick to several common fields at the beginning of each line. Identify time, severity, and subsystem in every line. Clearly formulate the message. Make every log message easy to map to its source code line. If an error happens, try to collect and log as much information as possible. It may take long but it's OK because normal processing has failed anyway. Not having to wait when the same condition happens in production with a debugger attached is priceless.

2.7 Lập trình hàm

Functional Without mutable variables, assignment, conditional

Advantages 1 Most functional languages provide a nice, protected environment, somewhat like JavaLanguage. It's good to be able to catch exceptions instead of having CoreDumps in stability-critical applications. FP encourages safe ways of programming. I've never seen an OffByOne mistake in a functional program, for example... I've seen one. Adding two lengths to get an index but one of them was zero-indexed. Easy to discover though. – AnonymousDonor Functional programs tend to be much more terse than their ImperativeLanguage counterparts. Often this leads to enhanced programmer productivity. FP encourages quick prototyping. As such, I think it is the best software design paradigm for ExtremeProgrammers... but what do I know. FP is modular in the dimension of functionality, where ObjectOrientedProgramming is modular in the dimension of different components. Generic routines (also provided by CeePlusPlus) with easy syntax. ParametricPolymorphism The ability to have your cake and eat it. Imagine you have a complex OO system processing messages - every component might make state changes depending on the message and then forward the message to some objects it has links to. Wouldn't it be just too cool to be able to easily roll back every change if some object deep in the call hierarchy decided the message is flawed? How about having a history of different states? Many housekeeping tasks made for you: deconstructing data structures (PatternMatching), storing variable bindings (LexicalScope with closures), strong typing (TypeInference), * GarbageCollection, storage allocation, whether to use boxed (pointer-to-value) or unboxed (value directly) representation... Safe multithreading! Immutable data structures are not subject to data race conditions, and consequently don't have to be protected by locks. If you are always allocating new objects, rather than destructively manipulating existing ones, the locking can be hidden in the allocation and GarbageCollection system.

2.8 Lập trình song song

Parallel/Concurrency Programming 1. Callback Pattern 2 Callback functions are derived from a programming paradigm known as functional programming. At a fundamental level, functional programming specifies the use of functions as arguments. Functional programming was—and still is, though to a much lesser extent today—seen as an esoteric technique of specially trained, master programmers.

Fortunately, the techniques of functional programming have been elucidated so that mere mortals like you and me can understand and use them with ease. One of the chief techniques in functional programming happens to be callback functions. As you will read shortly, implementing callback functions is as easy as passing regular variables as arguments. This technique is so simple that I wonder why it is mostly covered in advanced JavaScript topics.

```
[code lang="javascript"] function getN() return 10;
var n = getN();
function getAsyncN(callback) setTimeout(function() callback(10); , 1000);
function afterGetAsyncN(result) var n = 10; console.log(n);
getAsyncN(afterGetAsyncN); [/code]
```

2. Promise Pattern 1 3 What is a promise? The core idea behind promises is that a promise represents the result of an asynchronous operation.

A promise is in one of three different states:

pending - The initial state of a promise. fulfilled - The state of a promise representing a successful operation. rejected - The state of a promise representing a failed operation. Once a promise is fulfilled or rejected, it is immutable (i.e. it can never change again).

```
function aPromise(message){
  return new Promise(function(fulfill, reject){
    if(message == "success"){
      fulfill("it is a success Promise");
    } if(message == "fail"){
      reject("it is a fail Promise");
    }
  });
}
```

Usage:

```
aPromise("success").then(function(successMessage){
  console.log(successMessage) }, function(failMessage){
  // it is a success Promise
  console.log(failMessage)
})
```

```
aPromise("fail").then(function(successMessage){
  console.log(successMessage) }, function(failMessage){
  console.log(failMessage)
}) // it is a fail Promise
```

2.9 IDE

An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. Most modern IDEs have intelligent code completion.

1. Navigation

Word Navigation Line Navigation File Navigation

2. Editing

Auto Complete Code Complete Multicursor Template (Snippets)

3. Formatting

Debugging Custom Rendering for Object

Chương 3

Python

3.1 Giới thiệu

‘Python’ is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java.

The language provides constructs intended to enable clear programs on both a small and large scale.

Python Tutorial Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

Python is Interpreted

Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive

You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented

Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is Beginner Friendly

Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Audience This tutorial is designed for software programmers who need to learn Python programming language from scratch.

Sách

[Tập hợp các sách python](#)

Khoá học

[Tập hợp các khóa học python](#)

Tham khảo

Top 10 Python Libraries Of 2015

3.2 Cài đặt

Get Started Welcome! This tutorial details how to get started with Python.

For Windows Anaconda 4.3.0 Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.

1. Download the installer 2. Optional: Verify data integrity with MD5 or SHA-256 3. Double-click the .exe file to install Anaconda and follow the instructions on the screen Python 3.6 version 64-BIT INSTALLER Python 2.7 version 64-BIT INSTALLER Step 2. Discover the Map

<https://docs.python.org/2/library/index.html>

For CentOS Developer tools The Development tools will allow you to build and compile software from source code. Tools for building RPMs are also included, as well as source code management tools like Git, SVN, and CVS.

```
yum groupinstall "Development tools"
yum install zlib-devel
yum install bzip2-devel
yum install openssl-devel
yum install ncurses-devel
yum install sqlite-devel
```

Python Anaconda Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.

```
cd /opt
wget --no-check-certificate https://www.python.org/ftp/python
  ↪ /2.7.6/Python-2.7.6.tar.xz
tar xf Python-2.7.6.tar.xz
cd Python-2.7.6
./configure --prefix=/usr/local
make && make altinstall
## link
ln -s /usr/local/bin/python2.7 /usr/local/bin/python
# final check
which python
python -V
# install Anaconda
cd ~/Downloads
wget https://repo.continuum.io/archive/Anaconda-2.3.0-Linux-
  ↪ x86_64.sh
bash ~/Downloads/Anaconda-2.3.0-Linux-x86_64.sh
```

3.3 Cơ bản

3.4 Cú pháp cơ bản

Print, print

```
print "Hello World"
```

Conditional

```
if you_smart:
    print "learn python"
else:
    print "go away"
```

Loop

In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.

Programming languages provide various control structures that allow for more complicated execution paths. A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement

Python programming language provides following types of loops to handle looping requirements.

while loop Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body. **for loop** Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable. **nested loops** You can use one or more loop inside any another while, for or do..while loop. **While Loop** A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax

The syntax of a while loop in Python programming language is

```
while expression:
    statement(s)
```

Example

```
count = 0
while count < 9:
    print 'The count is:', count
    count += 1
print "Good bye!"
```

For Loop

It has the ability to iterate over the items of any sequence, such as a list or a string.

Syntax

```
for iterating_var in sequence:
    statements(s)
```

If a sequence contains an expression list, it is evaluated first. Then, the first item in the sequence is assigned to the iterating variable *iterating_var*. Next, the statements block is executed. Each time, the next item in the sequence is assigned to the iterating variable.

Example

```
for i in range(10):
    print "hello", i
```

```
for letter in 'Python':
    print 'Current letter :', letter
```

```
fruits = ['banana', 'apple', 'mango']
for fruit in fruits:
    print 'Current fruit :', fruit
```

```
print "Good bye!"
```

Yield and Generator

Yield is a keyword that is used like return, except the function will return a generator.

```
def createGenerator():
    yield 1
    yield 2
    yield 3
mygenerator = createGenerator() # create a generator
print(mygenerator) # mygenerator is an object!
# <generator object createGenerator at 0xb7555c34>
for i in mygenerator:
    print(i)
# 1
# 2
# 3
```

Visit [Yield and Generator](#) explained for more information
[Functions](#)
[Variable-length arguments](#)

```
def functionname([formal_args,] *var_args_tuple ):
    "function_docstring"
    function_suite
    return [expression]
```

Example

```
#!/usr/bin/python
```

```
# Function definition is here
def printinfo( arg1, *vartuple ):
    "This prints a variable passed arguments"
    print "Output is: "
    print arg1
    for var in vartuple:
        print var
    return;
```

```
# Now you can call printinfo function
printinfo( 10 )
printinfo( 70, 60, 50 )
```

Coding Convention Code layout Indentation: 4 spaces

Suggest Readings

"Python Functions". www.tutorialspoint.com "Python Loops". www.tutorialspoint.com
 "What does the "yield" keyword do?". stackoverflow.com "Improve Your Python:
 'yield' and Generators Explained". jeffknupp.com

Vấn đề với mảng

Random Sampling ¹ - sinh ra một mảng ngẫu nhiên trong khoảng (0, 1), mảng ngẫu nhiên số nguyên trong khoảng (x, y), mảng ngẫu nhiên là permutation của số từ 1 đến n

3.5 Yield and Generators

Coroutines and Subroutines When we call a normal Python function, execution starts at function's first line and continues until a return statement, exception, or the end of the function (which is seen as an implicit return None) is encountered. Once a function returns control to its caller, that's it. Any work done by the function and stored in local variables is lost. A new call to the function creates everything from scratch.

This is all very standard when discussing functions (more generally referred to as subroutines) in computer programming. There are times, though, when it's beneficial to have the ability to create a "function" which, instead of simply returning a single value, is able to yield a series of values. To do so, such a function would need to be able to "save its work," so to speak.

I said, "yield a series of values" because our hypothetical function doesn't "return" in the normal sense. return implies that the function is returning control of execution to the point where the function was called. "Yield," however, implies that the transfer of control is temporary and voluntary, and our function expects to regain it in the future.

In Python, "functions" with these capabilities are called generators, and they're incredibly useful. generators (and the yield statement) were initially introduced to give programmers a more straightforward way to write code responsible for producing a series of values. Previously, creating something like a random number generator required a class or module that both generated values and kept track of state between calls. With the introduction of generators, this became much simpler.

To better understand the problem generators solve, let's take a look at an example. Throughout the example, keep in mind the core problem being solved: generating a series of values.

Note: Outside of Python, all but the simplest generators would be referred to as coroutines. I'll use the latter term later in the post. The important thing to remember is, in Python, everything described here as a coroutine is still a generator. Python formally defines the term generator; coroutine is used in discussion but has no formal definition in the language.

Example: Fun With Prime Numbers Suppose our boss asks us to write a function that takes a list of ints and returns some Iterable containing the elements which are prime numbers.

¹tham khảo [pytorch](<http://pytorch.org/docs/master/torch.html?highlight=randntorch.randn>), [numpy](<https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.random.html>))

Remember, an Iterable is just an object capable of returning its members one at a time.

"Simple," we say, and we write the following:

```
def get_primes(input_list):
    result_list = list()
    for element in input_list:
        if is_prime(element):
            result_list.append()

    return result_list
```

or better yet...

```
def get_primes(input_list):
    return (element for element in input_list if is_prime(element))
    ↪ )
```

*# not germane to the example, but here's a possible
 ↪ implementation of
 # is_prime...*

```
def is_prime(number):
    if number > 1:
        if number == 2:
            return True
        if number % 2 == 0:
            return False
        for current in range(3, int(math.sqrt(number) + 1), 2):
            if number % current == 0:
                return False
        return True
    return False
```

Either *get_pprimes* implementation above fulfills the requirements, so we tell our boss we're done. She reports so. Dealing With Infinite Sequences Well, not quite exactly. A few days later, our boss comes back and tells us she's run into a small problem: she wants to use our *get_pprimes* function on a very large list of numbers. In fact, the list is so large that merely creating it would consume

Once we think about this new requirement, it becomes clear that it requires more than a simple change to *get_pprimes*. Clearly, we can't return a list of all the prime numbers from start to infinity.

Before we give up, let's determine the core obstacle preventing us from writing a function that satisfies our boss's new requirements. Thinking about it, we arrive at the following: functions only get one chance to return results, and thus must return all results at once. It seems pointless to make such an obvious statement; "functions just work that way," we think. The real value lies in asking, "but what if they didn't?"

Imagine what we could do if *get_pprimes* could simply return the next value instead of all the values at once. It would

Unfortunately, this doesn't seem possible. Even if we had a magical function that allowed us to iterate from *n* to infinity, we'd get stuck after returning the first value:

```
def get_pprimes(start) : for element in magical_infinite_range(start) : if is_prime(element) :
    return element
Imagine get_pprimes is called like so :
```

```
def solve_number10() : She*is*workingonProjectEuler10,Iknewit!total =
2fornext_primeinget_primes(3) : ifnext_prime < 2000000 : total+= next_primeelse :
print(total)returnClearly, inget_primes, we would immediately hit the case where number =
3and return at line 4. Instead of return, we need a way to generate a value and, when asked for the next one, pick up
```

Functions, though, can't do this. When they return, they're done for good. Even if we could guarantee a function would be called again, we have no way of saying, "OK, now, instead of starting at the first line like we normally do, start up where we left off at line 4." Functions have a single entry point: the first line.

Enter the Generator This sort of problem is so common that a new construct was added to Python to solve it: the generator. A generator "generates" values. Creating generators was made as straightforward as possible through the concept of generator functions, introduced simultaneously.

A generator function is defined like a normal function, but whenever it needs to generate a value, it does so with the yield keyword rather than return. If the body of a def contains yield, the function automatically becomes a generator function (even if it also contains a return statement). There's nothing else we need to do to create one.

generator functions create generator iterators. That's the last time you'll see the term generator iterator, though, since they're almost always referred to as "generators". Just remember that a generator is a special type of iterator. To be considered an iterator, generators must define a few methods, one of which is next(). To get the next value from a generator, we use the same built-in function as for iterators: next().

This point bears repeating: to get the next value from a generator, we use the same built-in function as for iterators: next().

(next() takes care of calling the generator's next() method). Since a generator is a type of iterator, it can be used in a for loop.

So whenever next() is called on a generator, the generator is responsible for passing back a value to whomever called next(). It does so by calling yield along with the value to be passed back (e.g. yield 7). The easiest way to remember what yield does is to think of it as return (plus a little magic) for generator functions.**

Again, this bears repeating: yield is just return (plus a little magic) for generator functions.

Here's a simple generator function:

```
>>> def simple_generator_function() :>>> yield1 >>> yield2 >>> yield3And here are two simple ways to use
>>> for value in simple_generator_function() :>>> print(value)123 >>>
our_generator = simple_generator_function() >>> next(our_generator)1 >>>
next(our_generator)2 >>> next(our_generator)3Magic?What's the magic part? Glad you asked! When a generator
Let's rewrite get_primes as a generator function. Notice that we no longer need the magical infinite range function
def get_primes(number) : while True : if is_prime(number) : yield number number += 1
If a generator function calls return or reaches the end of its definition, a StopIteration exception is raised. This signal
loop is presenting get_primes. If it weren't, the first time next() was called we would check if the number is prime and
>>> our_generator = simple_generator_function() >>> for value in our_generator :>>>
print(value)
```

```
>>> our_generator has been exhausted... >>> print(next(our_generator))Traceback (most recent call last) :
File "ipython - input - 13 - 7e48a609051a" > ", line 1, in < module >
next(our_generator)StopIteration
```

>>> however, we can always create a new generator >>> by calling the generator function again...

»> new_generator = simple_generator_function() >>> print(next(new_generator)) perfectly valid! Thus, the flow is as follows. Visualizing the flow. Let's go back to the code that was calling get_primes : solve_number_10.

```
def solve_number_10():
    She *is* working on Project Euler 10, I knew it! total = 2
    for next_prime in get_primes(3):
        if next_prime < 2000000:
            total += next_prime
        else:
            print(total)
            return
    It's helpful to visualize how the first few elements are created when we call get_primes in solve_number_10.
```

We enter the while loop on line 3. The if condition holds (3 is prime). We yield the value 3 and control to solve_number_10. Then, back in solve_number_10 :

```
The value 3 is passed back to the for loop. The for loop assigns next_prime to this value.
next_prime is added to total.
def get_primes(number):
    while True:
        if is_prime(number):
            yield number
            number += 1
    <<<<<<<<<< Most importantly, number still has the same value it did when we called yield (i.e. 3). Remember
```

Moar Power In PEP 342, support was added for passing values into generators. PEP 342 gave generators the power to yield a value (as before), receive a value, or both yield a value and receive a (possibly different) value in a single statement.

To illustrate how values are sent to a generator, let's return to our prime number example. This time, instead of simply printing every prime number greater than number, we'll find the smallest prime number greater than successive powers of a number (i.e. for 10, we want the smallest prime greater than 10, then 100, then 1000, etc.). We start in the same way as get_primes :

```
def print_successive_primes(iterations, base = 10):
    like normal functions, a generator function can be assigned to a variable.
    prime_generator = get_primes(base)
    missing code... for power in range(iterations):
```

```
def get_primes(number):
    while True:
        if is_prime(number):
            ... what goes here?
    Then the next line of get_primes is
    yield foo and, "yield foo and, when a value is sent to me, set to that value." You can "send" values to a generator.
```

```
def get_primes(number):
    while True:
        if is_prime(number):
            number =
            yield number
            number += 1
    In this way, we can set number to a different value each time the generator yields. We can
```

```
def print_successive_primes(iterations, base = 10):
    prime_generator = get_primes(base)
    prime_generator.send(base)
    print(prime_generator.send(base ** power))
    Two things to note here: First, we're reprinting the result of generator.send().
```

Second, notice the prime_generator.send(None) line. When you're using send to "start" a generator (that is, call it), you must pass None.

Round-up In the second half of this series, we'll discuss the various ways in which generators have been enhanced and the power they gained as a result. yield has become one of the most powerful keywords in Python. Now that we've built a solid understanding of how yield works, we have the knowledge necessary to understand some of the more "mind-bending" things that yield can be used for.

Believe it or not, we've barely scratched the surface of the power of yield. For example, while send does work as described above, it's almost never used when generating simple sequences like our example. Below, I've pasted a small demonstration of one common way send is used. I'll not say any more about it as figuring out how and why it works will be a good warm-up for part two.

```
import random
```

```
def get_data():
    """Return 3 random integers between 0 and 9"""
    return random.sample(range(10), 3)
```

```
def consume():
```



```

    """Displays a running average across lists of integers sent to
    ↪ it"""
    running_sum = 0
    data_items_seen = 0

    while True:
        data = yield
        data_items_seen += len(data)
        running_sum += sum(data)
        print('The running average is {}'.format(running_sum /
        ↪ float(data_items_seen)))

def produce(consumer):
    """Produces a set of values and forwards them to the pre-
    ↪ defined consumer
    function"""
    while True:
        data = get_data()
        print('Produced {}'.format(data))
        consumer.send(data)
        yield

if __name__ == '__main__':
    consumer = consume()
    consumer.send(None)
    producer = produce(consumer)

    for _ in range(10):
        print('Producing...')
        next(producer)

```

Remember... There are a few key ideas I hope you take away from this discussion:

generators are used to generate a series of values yield is like the return of generator functions The only other thing yield does is save the "state" of a generator function A generator is just a special type of iterator Like iterators, we can get the next value from a generator using next() for gets values by calling next() implicitly

3.6 Cấu trúc dữ liệu

Number Basic Operation

```

1
1.2
1 + 2
abs(-5)

```

3.7 Quản lý gói với Anaconda

Cài đặt package tại một branch của một project trên github

```
$ pip install git+https://github.com/tangentlabs/django-oscar-
  ↳ paypal.git@issue/34/oscar-0.6#egg=django-oscar-paypal
```

Trích xuất danh sách package

```
$ pip freeze > requirements.txt
```

Chạy ipython trong environment anaconda

Chạy dòng lệnh này

```
conda install nb_conda
source activate my_env
python -m IPython kernelspec install-self --user
ipython notebook
```

Interactive programming với ipython

Trích xuất ipython ra slide (không hiểu sao default ‘--to slides’ không work nữa, lại phải thêm tham số ‘--reveal-prefix’ [1])

```
jupyter nbconvert "file.ipynb"
--to slides
--reveal-prefix "https://cdnjs.cloudflare.com/ajax/libs/reveal.
  ↳ js/3.1.0"
```

****Tham khảo thêm****

* <https://stackoverflow.com/questions/37085665/in-which-conda-environment-is-jupyter-executing> * <https://github.com/jupyter/notebook/issues/541#issuecomment-146387578> * <https://stackoverflow.com/a/20101940/772391>

python 3.4 hay 3.5

Có lẽ 3.5 là lựa chọn tốt hơn (phải có của tensorflow, pytorch, hỗ trợ mock)

Quản lý môi trường phát triển với conda

Chạy lệnh ‘remove’ để xóa một môi trường

```
conda remove --name flowers --all
```

3.8 Test với python

Sử dụng những loại test nào?

Hiện tại mình đang viết unittest với default class của python là unittest. Thực ra toàn sử dụng ‘assertEqual’ là chính!

Ngoài ra mình cũng đang sử dụng tox để chạy test trên nhiều phiên bản python (python 2.7, 3.5). Điều hay của tox là mình có thể thiết kế toàn bộ cài đặt project và các dependencies package trong file ‘tox.ini’

Chạy test trên nhiều phiên bản python với tox

Pycharm hỗ trợ debug tox (quá tuyệt!), chỉ với thao tác đơn giản là nhấn chuột phải vào file tox.ini của project.

3.9 Xây dựng docs với readthedocs và sphinx

20/12/2017: Tự nhiên hôm nay tất cả các class có khai báo kế thừa ở project languageflow không thể index được. Vãi thật. Làm thằng đệ không biết đâu mà build model.

Thử build lại chục lần, thay đổi file conf.py và package_reference.rst chán chê không được. Giả thiết đầu tiên là do hai nguyên nhân (1) docstring ghi sai, (2) nội dung trong package_reference.rst bị sai. Sửa chán chê cũng vẫn thế, thử checkout các commit của git. Không hoạt động!

Mất khoảng vài tiếng mới để ý thằng readthedocs có phần log cho từng build một. Lần mò vào build gần nhất và build (mình nhớ là) thành công cách đây 2 ngày

Log build gần nhất

```
Running Sphinx v1.6.5
making output directory...
loading translations [en]... done
loading intersphinx inventory from https://docs.python.org/
  ↳ objects.inv...
intersphinx inventory has moved: https://docs.python.org/objects.
  ↳ inv -> https://docs.python.org/2/objects.inv
loading intersphinx inventory from http://docs.scipy.org/doc/
  ↳ numpy/objects.inv...
intersphinx inventory has moved: http://docs.scipy.org/doc/numpy/
  ↳ objects.inv -> https://docs.scipy.org/doc/numpy/objects.
  ↳ inv
building [mo]: targets for 0 po files that are out of date
building [readthedocsdirhtml]: targets for 8 source files that
  ↳ are out of date
updating environment: 8 added, 0 changed, 0 removed
reading sources... [ 12%] authors
reading sources... [ 25%] contributing
reading sources... [ 37%] history
reading sources... [ 50%] index
reading sources... [ 62%] installation
reading sources... [ 75%] package_reference
reading sources... [ 87%] readme
reading sources... [100%] usage

looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [ 12%] authors
writing output... [ 25%] contributing
writing output... [ 37%] history
writing output... [ 50%] index
writing output... [ 62%] installation
writing output... [ 75%] package_reference
writing output... [ 87%] readme
```

```
writing output... [100%] usage

    Log build hồi trước

Running Sphinx v1.5.6
making output directory...
loading translations [en]... done
loading intersphinx inventory from https://docs.python.org/
    ↪ objects.inv...
intersphinx inventory has moved: https://docs.python.org/objects.
    ↪ inv -> https://docs.python.org/2/objects.inv
loading intersphinx inventory from http://docs.scipy.org/doc/
    ↪ numpy/objects.inv...
intersphinx inventory has moved: http://docs.scipy.org/doc/numpy/
    ↪ objects.inv -> https://docs.scipy.org/doc/numpy/objects.
    ↪ inv
building [mo]: targets for 0 po files that are out of date
building [readthedocs]: targets for 8 source files that are out
    ↪ of date
updating environment: 8 added, 0 changed, 0 removed
reading sources... [ 12%] authors
reading sources... [ 25%] contributing
reading sources... [ 37%] history
reading sources... [ 50%] index
reading sources... [ 62%] installation
reading sources... [ 75%] package_reference
reading sources... [ 87%] readme
reading sources... [100%] usage

/home/docs/checkouts/readthedocs.org/user_builds/languageflow/
    ↪ checkouts/develop/languageflow/transformer/count.py:
    ↪ docstring of languageflow.transformer.count.
    ↪ CountVectorizer:106: WARNING: Definition list ends without
    ↪ a blank line; unexpected unindent.
/home/docs/checkouts/readthedocs.org/user_builds/languageflow/
    ↪ checkouts/develop/languageflow/transformer/tfidf.py:
    ↪ docstring of languageflow.transformer.tfidf.
    ↪ TfidfVectorizer:113: WARNING: Definition list ends without
    ↪ a blank line; unexpected unindent.
../README.rst:7: WARNING: nonlocal image URI found: https://img.
    ↪ shields.io/badge/latest-1.1.6-brightgreen.svg
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [ 12%] authors
writing output... [ 25%] contributing
writing output... [ 37%] history
writing output... [ 50%] index
writing output... [ 62%] installation
writing output... [ 75%] package_reference
```

```
writing output... [ 87%] readme
writing output... [100%] usage
```

Đập vào mắt là sự khác biệt giữa documentation type
Lỗi

```
building [readthedocsdirhtml]: targets for 8 source files that
↳ are out of date
```

Chạy

```
building [readthedocs]: targets for 8 source files that are out
↳ of date
```

Hí ha hí hửng. Chắc trong cơn bất loạn sửa lại settings đây mà. Sửa lại nó trong phần Settings (Admin gt; Settings gt; Documentation type)

Khi chạy nó đã cho ra log đúng

```
building [readthedocsdirhtml]: targets for 8 source files that
↳ are out of date
```

Nhưng vẫn lỗi. Vãi!!! Sau khoảng 20 phút tiếp tục bắn loạn, chửi bới readthedocs các kiểu. Thì để ý dòng này
Lỗi

Running Sphinx v1.6.5

Chạy

Running Sphinx v1.5.6

Ngay dòng đầu tiên mà không để ý, ngu thật. Aha, Hóa ra là thằng readthedocs nó tự động update phiên bản sphinx lên 1.6.5. Mình là mình chúa ghét thay đổi phiên bản (code đã mệt rồi, lại còn phải tương thích với nhiều phiên bản nữa thì ăn c** à). Đầu tiên search với Pycharm thấy dòng này trong ‘conf.py’

```
# If your documentation needs a minimal Sphinx version, state it
↳ here.
# needs_sphinx = '1.0'
```

Đổi thành

```
# If your documentation needs a minimal Sphinx version, state it
↳ here.
needs_sphinx = '1.5.6'
```

Vẫn vậy (holy sh*t). Thử sâu một tạo (thực sự là rất nhiều tạo). Thấy cái này trong trang Settings

Ồ há. Thằng đàn này cho phép trở đường dẫn tới một file trong project để cấu hình dependency. Haha. Tạo thêm một file ‘requirements’ trong thư mục ‘docs’ với nội dung

```
sphinx==1.5.6
```

Sau đó cấu hình nó trên giao diện web của readthedocs

Build thử. Build thử thôi. Cảm giác đúng lắm rồi đấy. Và... nó chạy. Ahihi

Kinh nghiệm

* Khi không biết làm gì, hãy làm 3 việc. Đọc LOG. Phân tích LOG. Và cố gắng để LOG thay đổi theo ý mình.

PS: Trong quá trình này, cũng không thêm build thẳng PDF với Epub nữa. Tiết kiệm được bao nhiêu thời gian.

3.10 Pycharm Pycharm

01/2018: Pycharm là trình duyệt ưa thích của mình trong suốt 3 năm vừa rồi.

Hôm nay tự nhiên lại gặp lỗi không tự nhận unittest, không resolve được package import bởi relative path. Vụ không tự nhận unittest sửa bằng cách xóa file .idea là xong. Còn vụ không resolve được package import bởi relative path thì vẫn chịu rồi. Nhìn code cứ đồ lờm khó chịu thật.

3.11 Vì sao lại code python?

01/11/2017 Thích python vì nó quá đơn giản (và quá đẹp).

[¹] : <https://github.com/jupyter/nbconvert/issues/91#issuecomment-283736634>

Chương 4

Java

01/11/2017: Java đơn giản là gay nhé. Không chơi. Viết java chỉ viết thế này thôi. Không viết hơn. Thề!

Chương 5

PHP

PHP là ngôn ngữ lập trình web dominate tất cả các anh tài khác mà (chắc là) chỉ dụi đi khi mô hình REST xuất hiện. Nhớ lần đầu gặp bạn Laravel mà cảm giác cuộc đời sang trang.

Cuối tuần này lại phải xem làm sao cài được xdebug vào PHPStorm cho thằng em tập tành lập trình. Haizzz

Tương tác với cơ sở dữ liệu

Liệt kê danh sách các bản ghi trong bảng groups

```
“sql = "SELECT * FROM 'groups'";groups = mysqli_query(conn, sql);“
```

Xóa một bản ghi trong bảng groups

```
“sql = "DELETE FROM 'groups' WHERE Id = '5'";mysqli_query(conn, sql);“
```

Cài đặt debug trong PHPStorm

<https://www.youtube.com/watch?v=mEJ21RB0F14>

(1) XAMPP

- Download XAMPP (cho PHP 7.1.x - do XDebug chưa chính thức hỗ trợ 7.2.0) <https://www.apachefriends.org/xampp-files/7.1.12/xampp-win32-7.1.12-0-VC14-installer.exe> - Install XAMPP xampp-win32-7.1.12-0-VC14-installer.exe
- Truy cập vào địa chỉ <http://localhost/dashboard/phpinfo.php> để kiểm tra cài đặt đã thành công chưa

(2) Tải và cài đặt PHPStorm

- Download PHPStorm <https://download-cf.jetbrains.com/webide/PhpStorm-2017.3.2.exe> - Install PHPStorm

(3) Tạo một web project trong PHPStorm - Chọn interpreter trở đến PHP trong xampp

(4) Viết một chương trình add.php

```
“php a = 2; b = 3; c = a + b;
```

```
echo c;“
```

Click vào ‘add.php’, chọn Debug, PHPStorm sẽ báo chưa cài XDebug

(5) Cài đặt XDebug theo hướng dẫn tại <https://gist.github.com/odan/1abe76d373a9cbb15bed>

Click vào add.php, chọn Debug

(6) Cài đặt XDebug với PHPStorm Marklets Vào trang <https://www.jetbrains.com/phpstorm/marklets/>

Trong phần Zend Debugger - chọn cổng 9000 - IP: 127.0.0.1 Nhấn nút Generate

Bookmark các link `Start debugger`, `Stop debugger`; lên trình duyệt

(7) Debug PHP từ trình duyệt

* Vào trang <http://localhost/untitled/add.php> * Click vào bookmark Start debugger * Trong PHPStorm, nhấn vào biểu tượng `Start Listening for PHP Debug Connections`; * Đặt breakpoint tại dòng thứ 5 * Refresh lại trang <http://localhost/untitled/add.php>, lúc này, breakpoint sẽ dừng ở dòng 5

Phần II

Xác suất

Chương 6

Các hàm phân phối thông dụng

Phần này có thêm khảo [Goodfellow u.a. \(2016\)](#) và giáo trình xác suất thống kê của thạc sỹ Trần Thiện Khải, đại học Trà Vinh ¹

17/01/2018 Lòng vòng thế nào hôm nay lại tìm được của bạn Đỗ Minh Hải ², rất hay

6.0.1 Biến rời rạc

Phân phối đều - Discrete Uniform distribution

Là phân phối mà xác suất xuất hiện của các sự kiện là như nhau.
Biến ngẫu nhiên X tuân theo phân phối đều rời rạc

$$X \sim \mathcal{U}(a, b)$$

với tham số $a, b \in \mathbb{Z}; a < b$ là khoảng giá trị của X , đặt $n = b - a + 1$

Ta sẽ có:

Định nghĩa	Giá trị
PMF	$p(x) \mid \frac{1}{n}, \forall x \in [a, b]$
CDF - $F(x; a, b)$	$\frac{x - a + 1}{n}, \forall x \in [a, b]$
Kỳ vọng - $E[X]$	$\frac{a + b}{2}$
Phương sai - $Var(X)$	$\frac{n^2 - 1}{12}$

Ví dụ: Lịch chạy của xe buýt tại một trạm xe buýt như sau: chiếc xe buýt đầu tiên trong ngày sẽ khởi hành từ trạm này vào lúc 7 giờ, cứ sau mỗi 15 phút sẽ có một xe khác đến trạm. Giả sử một hành khách đến trạm trong khoảng thời gian từ 7 giờ đến 7 giờ 30. Tìm xác suất để hành khách này chờ:

- Ít hơn 5 phút.
- Ít nhất 12 phút.

Giải

¹http://www.ctec.tvu.edu.vn/ttkhai/xacsuatthongke_dh.htm

²<https://dominhhai.github.io/vi/2017/10/prob-com-var>

Gọi X là số phút sau 7 giờ mà hành khách đến trạm.

Ta có: $X \sim R[0; 30]$.

a) Hành khách sẽ chờ ít hơn 5 phút nếu đến trạm giữa 7 giờ 10 và 7 giờ 15 hoặc giữa 7 giờ 25 và 7 giờ 30. Do đó xác suất cần tìm là:

$$P(0 < X < 15) + P(25 < X < 30) = \frac{5}{30} + \frac{5}{30} = \frac{1}{3}$$

b) Hành khách chờ ít nhất 12 phút nếu đến trạm giữa 7 giờ và 7 giờ 3 phút hoặc giữa 7 giờ 15 phút và 7 giờ 18 phút. Xác suất cần tìm là:

$$P(0 < X < 3) + P(15 < X < 18) = \frac{3}{30} + \frac{3}{30} = \frac{1}{5}$$

Phân phối Béc-nu-li - Bernoulli distribution

Như đã đề cập về phép thử Béc-nu-li rằng mọi phép thử của nó chỉ cho 2 kết quả duy nhất là A với xác suất p và \bar{A} với xác suất $q = 1 - p$. Biến ngẫu nhiên X tuân theo phân phối Béc-nu-li

$$X \sim B(p)$$

với tham số $p \in \mathbb{R}, 0 \leq p \leq 1$ là xác suất xuất hiện của A tại mỗi phép thử

Định nghĩa		Giá trị
PMF	$p(x)$	$p(x) \mid p^x(1-p)^{1-x}, x \in \{0, 1\}$
CDF	$F(x; p)$	$\begin{cases} 0 & \text{for } x < 0 \\ 1-p & \text{for } 0 \leq x < 1 \\ 1 & \text{for } x \geq 1 \end{cases}$
Kỳ vọng	$E[X]$	p
Phương sai	$Var(X)$	$p(1-p)$

Ví dụ

Tham khảo thêm các thuật toán khác tại [Hai \(2018\)](#)

Phần III

Khoa học máy tính

Chương 7

Hệ điều hành

Những phần mềm không thể thiếu

* Trình duyệt Google Chrome (với các extensions Scihub, Mendeley Desktop, Adblock) * Adblock extension * Terminal (Oh-my-zsh) * IDE Pycharm để code python * Quản lý phiên bản code Git * Bộ gõ ibus-unikey trong Ubuntu hoặc unikey (Windows) (Ctrl-Space để chuyển đổi ngôn ngữ) * CUDA (lập trình trên GPU)

****Xem thông tin hệ thống****

Phiên bản ‘ubuntu 16.04’

```
sudo apt-get install sysstat
```

Xem hoạt động (

“ mpstat -A “

CPU của mình có bao nhiêu core, bao nhiêu siblings

“ cat /proc/cpuinfo

processor : 23 vendor_id : GenuineIntelcpu family : 6model : 62modelname :

Intel(R) Xeon(R) CPU E5-2430v2@2.50GHzstepping : 4microcode : 0x428cpu MHz :

1599.707cachesize : 15360KBphysicalid : 1siblings : 12coreid : 5cpucore : 5

6apicid : 43initialapicid : 43fpu : yesfpu_exception : yescpuidlevel : 13wp :

yesflags : fpuvmdepsetscmsrpaemccecx8apicsepmttrrgemcacrmoovpatpse36clflushdtsacpimxfxsrssesse

5005.20clflushsize : 64cache_alignment : 64addresssizes : 46bitsphysical, 48bitsvirtualpowermanagement :

“

Kết quả cho thấy cpu của 6 core và 12 siblings

Chương 8

Ubuntu

****Chuyện terminal****

Terminal là một câu chuyện muôn thưở của bất kì ông coder nào thích customize, đẹp, tiện (và bug kinh hoàng). Hiện tại mình đang thấy combo này khá ổn Terminal (Ubuntu) (Color: Black on white, Build-in schemes: Tango) + zsh + oh-my-zsh (fishy-custom theme). Những features hay ho

* Làm việc tốt trên cả Terminal (white background) và embedded terminal của Pycharm (black background) * Hiển thị folder dạng ngắn (chỉ ký tự đầu tiên) * Hiển thị branch của git ở bên phải

![Imgur](https://i.imgur.com/q53vQdH.png)

****Chuyện bộ gõ****

Làm sao để khởi động lại ibus, thỉnh thoảng lại chết bất đắc kì tử ^[1] “ibus – daemonibusrestart”

****Chuyện lỗi login loop****

Phiên bản: ‘ubuntu 16.04’

27/12/2017: Lại dính lỗi không thể login. Lần này thì lại phải xóa bạn KDE đi. Kể cũng hơn buồn. Nhưng nhất quyết phải enable được tính năng Windows Spreading (hay đại loại thế). Hóa ra khi ubuntu bị lỗi không có launcher hay toolbar là do bạn unity plugin chưa được enable. Oái. Sao người hiền lành như mình suốt ngày bị mấy lỗi vớ vẩn thế không biết.

20/11/2017: Hôm nay đen thật, dính lỗi login loop. Fix mãi mới được. Thôi cũng kệ. Cảm giác bạn KDE này đỡ bị lỗi ibus-unikey hơn bạn GNOME. Hôm nay cũng đổi bạn zsh theme. Chọn mãi chẳng được bạn nào ổn ổn, nhưng không thể chịu được kiểu suggest lỗi nữa rồi. Đôi khi thấy default vẫn là tốt nhất.

21/11/2017: Sau một ngày trải nghiệm KDE, cảm giác giao diện mượt hơn GNOME. Khi overview windows với nhiều màn hình tốt và trực quan hơn. Đặc biệt là không bị lỗi ibus nữa. Đổi terminal cũng cảm giác ổn ổn. Không bị lỗi suggest nữa.

^[1] : <https://askubuntu.com/questions/389903/ibus-doesnt-seem-to-restart>

Phần IV

Khoa học dữ liệu

Chương 9

Học máy

- Vấn đề với HMM và CRF?
- Học MLE và MAP?

****Có bao nhiêu thuật toán Machine Learning?***

Có rất nhiều thuật toán Machine Learning, bài viết [Điểm qua các thuật toán Machine Learning hiện đại](<https://ongxuanhong.wordpress.com/2015/10/22/diem-qua-cac-thuat-toan-machine-learning-hien-dai/>) của Ông Xuân Hồng tổng hợp khá nhiều thuật toán. Theo đó, các thuật toán Machine Learning được chia thành các nhánh lớn như ‘regression’, ‘bayesian’, ‘regularization’, ‘decision tree’, ‘instance based’, ‘dimensionality reduction’, ‘clustering’, ‘deep learning’, ‘neural networks’, ‘associated rule’, ‘ensemble’... Ngoài ra thì còn có các cheatsheet của [sklearn](http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html).

Việc biết nhiều thuật toán cũng giống như ra đường mà có nhiều lựa chọn về xe cộ. Tuy nhiên, quan trọng là có task để làm, sau đó thì cập nhật SOTA của task đó để biết các công cụ mới.

****Xây dựng model cần chú ý điều gì?***

Khi xây dựng một model cần chú ý đến vấn đề tối ưu hóa tham số (có thể sử dụng [GridSearchCV]([sklearn.model_selection.GridSearchCV](https://sklearn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html)))

Bài phát biểu này có vẻ cũng rất hữu ích [PYCON UK 2017: Machine learning libraries you’d wish you’d known about](<https://www.youtube.com/watch?v=nDF78FOhpI>). *CÓ CP ON*

* [DistrictDataLabs/yellowbrick](<https://github.com/DistrictDataLabs/yellowbrick>) (giúp visualize model được train bởi sklearn) * [marcotcr/lime](<https://github.com/marcotcr/lime>) (giúp inspect classifier) * [TeamHG-Memex/eli5](<https://github.com/TeamHG-Memex/eli5>) (cũng giúp inspect classifier, hỗ trợ nhiều model như xgboost, crfsuite, đặc biệt có TextExplainer sử dụng thuật toán từ eli5) * [rhiever/tpot](<https://github.com/rhiever/tpot>) (giúp tối ưu hóa pipeline) * [dask/dask](<https://github.com/dask/dask>) (tính toán song song và lập lịch)

Ghi chú về các thuật toán trong xử lý ngôn ngữ tự nhiên tại [underthesea.flow/wiki](<https://github.com/magizbox/underthesea.flow/wiki/Develop>)

Framework để train, test hiện tại vẫn rất thoải mái sklearn. tensorboard cung cấp phần log cũng khá hay.

[Câu trả lời hay](<https://www.quora.com/What-are-the-most-important-machine-learning-techniques-to-master-at-this-time/answer/Sean-McClure-3?srid=5O2u>)

cho câu hỏi [Những kỹ thuật machine learning nào quan trọng nhất để master?](<https://www.quora.com/What-are-the-most-important-machine-learning-techniques-to-master-at-this-time>), đặc biệt là dẫn đến bài [The State of ML and Data Science 2017](<https://www.kaggle.com/surveys/2017>) của Kaggle.

****Tài liệu học PGM****

[Playlist youtube](<https://www.youtube.com/watch?v=WPSQfOkb1M8&list=PL50E6E80E8525B59C>) khóa học Probabilistic Graphical Models của cô Daphne Koller. Ngoài ra còn có một [tutorial](<http://mensxmachina.org/files/software/demos/bayesnetdemo.html>) dễ hơi ở đầu về tạo Bayesian network

****[Chưa biết] Tại sao Logistic Regression lại là Linear Model?***

Trong quyển Deep Learning, chương 6, trang 165, tác giả có viết

““ Linear models, such as logistic regression and linear regression, are appealing because they can be efficiently and reliably, either in closed form or with convex optimization ““

Mình tự hỏi tại sao logistic regression lại là linear, trong khi nó có sử dụng hàm logit (nonlinear)? Tìm hiểu hóa ra cũng có bạn hỏi giống mình trên [stats.stackexchange.com](<https://stats.stackexchange.com/questions/93569/why-is-logistic-regression-a-linear-classifier>). Ngoài câu trả lời trên stats.stackexchange, đọc một số cái khác [Generalized Linear Models, SPSS Statistics 22.0.0](https://www.ibm.com/support/knowledgecenter/IT23241_22.0.0/IntroductiontoGeneralizedLinearModels_AnalysisofDiscreteData_PennsylvaniaStateUniversity](<https://onlinecourses.science.psu.edu/stat504/node/216>)*engvnchahiulm*.

Hiện tại chỉ hiểu là các lớp model này chỉ có thể hoạt động trên các tập linear separable, có lẽ do việc map input x , luôn có một liên kết linear $latex wx$, trước khi đưa vào hàm non-linear.

****Các tập dữ liệu thú vị****

Iris dataset: dữ liệu về hoa iris

Là một ví dụ cho bài toán phân loại

Weather problem: dữ liệu thời tiết. Có thể tìm được ở trong quyển Data

Mining: Practical Machine Learning Tools and Techniques

Là một ví dụ cho bài toán cây quyết định

Deep Learning

****Tài liệu Deep Learning****

Lang thang thế nào lại thấy trang này [My Reading List for Deep Learning!](https://www.microsoft.com/en-us/research/wp-content/uploads/2017/02/DL_Reading_List.pdf)*camtanhMicrosoft.TrongO, (Ongnhin)cDee*

Các layer trong deep learning [2]

Sparse Layers

****nn.Embedding****(<http://pytorch.org/docs/master/nn.html#embedding>) ([hướng dẫn](http://pytorch.org/tutorials/beginner/nlp/word_embeddings_tutorial.html))*grepcode* : [Shawn1993/cnn-text-classification-pytorch](<https://github.com/Shawn1993/cnn-text-classification-pytorch/blob/master/model.py#L18>)*(Engvaitrnhmtlookuptable, mapmtwordvidenseve*


Convolution Layers

****nn.Conv1d****(<http://pytorch.org/docs/master/nn.html#conv1d>), ****nn.Conv2d****(<http://pytorch.org/docs/master/nn.html#conv2d>), ****nn.Conv3d****(<http://pytorch.org/docs/master/nn.html#conv3d>) [1]*grepcode* : [Shawn1993/cnn-text-classification-pytorch](<https://github.com/Shawn1993/cnn-text-classification-pytorch/blob/master/model.py#L20-L24>), [galsang/CNN-sentence-classification-pytorch](<https://github.com/galsang/CNN-sentence-classification-pytorch/blob/master/model.py#L36-L38>)

Các tham số trong Convolution Layer

* `kernel_size` (*hay* `filter_size`)

Đối với NLP, `kernel_sizethnbnregion_size*word_dim` (*Oiviconv1d*) *hay* (`region_size, word_dim`) *Oiviconv2d*

Quá trình tạo feature map đối với region size bằng 2
 (<https://media.giphy.com/media/l2QE2y1UQP7vIgit/giphy.gif>)
 * `'in_channels', 'out_channels' (labeled 'featuremaps')`

Kênh (channels) là các cách nhìn (view) khác nhau đối với dữ liệu. Ví dụ, trong ảnh thường có 3 kênh RGB (red, green, blue), có thể áp dụng convolution giữa các kênh. Với văn bản cũng có thể có các kênh khác nhau, như khi có các kênh sử dụng các word embedding khác nhau (word2vec, GloVe), hoặc cùng một câu nhưng biểu diễn ở các ngôn ngữ khác nhau.

* `'stride'`

Định nghĩa bước nhảy của filter.

 (<http://d3kbpzbmcyntmx.cloudfront.net/wp-content/uploads/2015/11/Screenshot-2015-11-05-at-10.18.08-AM-1024x251.png>)

Hình minh họa sự khác biệt giữa các feature map đối với `stride=1` và `stride=2`. Feature map đối với `stride = 1` có kích thước là 5, feature map đối với `stride = 3` có kích thước là 3. Stride càng lớn thì kích thước của feature map càng nhỏ.

Trong bài báo của Kim 2014, `'stride = 1'` đối với `'nn.conv2d'` và `'stride = word_dim'` đối với `'nn.conv1d'`

Toàn bộ tham số của mạng CNN trong bài báo Kim 2014,

 (<http://d3kbpzbmcyntmx.cloudfront.net/wp-content/uploads/2015/11/Screenshot-2015-11-06-at-8.03.47-AM.png>)

Description	Values
Google word2vec	filter region size (3, 4, 5) feature maps 100
activation function	ReLU pooling 1-max pooling dropout rate 0.5
$latex \lambda; s = 22$ norm constraint	3

Đọc thêm:

* [Lecture 13: Convolutional Neural Networks (for NLP). CS224n-2017] (<http://web.stanford.edu/class/cs224n-2017-lecture13-CNNs.pdf>) * [DeepNLP-models-Pytorch - 8. Convolutional Neural Networks] (<https://nbviewer.jupyter.org/github/DSKSD/DeepNLP-models-Pytorch/blob/master/notebooks/08.CNN-for-Text-Classification.ipynb>) * [A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. Zhang 2015] (<https://arxiv.org/pdf/1510.03820.pdf>)

****BTS****

22/11/2017 - Phải nói quyển này hơi nặng so với mình. Nhưng thôi cứ cố gắng vậy. 24/11/2017 - Từ hôm nay, mỗi ngày sẽ ghi chú một phần (rất rất nhỏ) về Deep Learning [tại đây] (https://docs.google.com/document/d/1KxDrw5s6uYHNLda7t0rhp0RM_TlUGxydQ-Qi1JOPFr8/edit?usp=sharing)

[¹] : [Understanding Convolutional Neural Networks for NLP] (<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp>) [²] : [<http://pytorch.org/docs/master/nn.html>] (<http://pytorch.org/docs/master/nn.html>)

Chương 10

Học sâu

10.1 Tài liệu Deep Learning

Lang thang thế nào lại thấy trang này [My Reading List for Deep Learning!](#) của một anh ở Microsoft. Trong đó, (đương nhiên) có Deep Learning của thánh Yoshua Bengio, có một vụ hay nữa là bài review "Deep Learning" của mấy thánh Yann Lecun, Yoshua Bengio, Geoffrey Hinton trên tạp chí Nature. Ngoài ra còn có nhiều tài liệu hữu ích khác.

10.2 Các layer trong deep learning

10.2.1 Sparse Layers

[nn.Embedding](#) (hướng dẫn)

grep code: [Shawn1993/cnn-text-classification-pytorch](#)

Đóng vai trò như một lookup table, map một word với dense vector tương ứng

10.2.2 Convolution Layers

[nn.Conv1d](#), [nn.Conv2d](#), [nn.Conv3d](#))

grep code: [Shawn1993/cnn-text-classification-pytorch](#), [galsang/CNN-sentence-classification-pytorch](#)

Các tham số trong Convolution Layer

* *kernel_size* (hay là filter size)

Đối với NLP, *kernel_size* thường bằng *region_size * word_dim* (đối với conv1d) hay (*region_size*, *word_dim*) đối với conv2d

<small>Quá trình tạo feature map đối với region size bằng 2</small>

* *'in_channels'*, *'out_channels'* (*lslnsg'featuremaps'*)

Kênh (channels) là các cách nhìn (view) khác nhau đối với dữ liệu. Ví dụ, trong ảnh thường có 3 kênh RGB (red, green, blue), có thể áp dụng convolution giữa các kênh. Với văn bản cũng có thể có các kênh khác nhau, như khi có các kênh sử dụng các word embedding khác nhau (word2vec, GloVe), hoặc cùng một câu nhưng biểu diễn ở các ngôn ngữ khác nhau.

* 'stride'

Định nghĩa bước nhảy của filter.

Hình minh họa sự khác biệt giữa các feature map đối với stride=1 và stride=2. Feature map đối với stride = 1 có kích thước là 5, feature map đối với stride = 3 có kích thước là 3. Stride càng lớn thì kích thước của feature map càng nhỏ.

Trong bài báo của Kim 2014, 'stride = 1' đối với 'nn.conv2d' và 'stride = word_{dim}' đối với 'nn.conv1d'

Toàn bộ tham số của mạng CNN trong bài báo Kim 2014,

Description	Values
Google word2vec	input word vectors
filter region size	(3, 4, 5)
feature maps	100
activation function	ReLU
pooling	1-max pooling
dropout rate	0.5
l ₂ norm constraint	3

Đọc thêm:

* [Lecture 13: Convolutional Neural Networks (for NLP). CS224n-2017](<http://web.stanford.edu/class/cs224n-2017-lecture13-CNNs.pdf>) * [DeepNLP-models-Pytorch - 8. Convolutional Neural Networks](<https://nbviewer.jupyter.org/github/DSKSD/DeepNLP-models-Pytorch/blob/master/notebooks/08.CNN-for-Text-Classification.ipynb>) * [A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. Zhang 2015](<https://arxiv.org/pdf/1510.03820.pdf>)

BTS

22/11/2017 - Phải nói quyển này hơi nặng so với mình. Nhưng thôi cứ cố gắng vậy. 24/11/2017 - Từ hôm nay, mỗi ngày sẽ ghi chú một phần (rất rất nhỏ) về Deep Learning [tại đây](https://docs.google.com/document/d/1KxDrw5s6uYHNLda7t0rhp0RM_TlUGxydQ-Qi1JOPFr8/edit?usp=sharing)

[¹]: [Understanding Convolutional Neural Networks for NLP](<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp>) [²]: <http://pytorch.org/docs/master/nn.html>

Chương 11

Xử lý ngôn ngữ tự nhiên

****05/01/2018****: "điên đầu" với Sphinx và HTK

HTK thì đã bỏ rồi vì quá lằng nhằng.

Sphinx thì setup được đối với dữ liệu nhỏ rồi. Nhưng không thể làm nó hoạt động với dữ liệu của VIVOS. Chắc hôm nay sẽ switch sang Kaldi vậy.

****26/12/2017****: Automatic Speech Recognition 100

Sau mấy ngày "vật lộn" với code base của Truong Do, thì cuối cùng cũng produce voice được. Cảm giác rất thú vị. Quyết định làm luôn ASR. Tìm mãi chẳng thấy code base đâu (chắc do lĩnh vực mới nên không có kinh nghiệm). May quá lại có bạn frankydotid có project về nhận diện tiếng Indonesia ở [github](https://github.com/frankydotid/Indonesian-Speech-Recognition). Trong README.md bạn đấy bảo là phải cần đọc HTK Book. Tốt quá đang cần cơ bản.

****20/12/2017****: Text to speech 100

Cảm ơn project rất hay của [bạn Truong Do ở vais](https://vais.vn/vi/tai-ve/hts_for_vietnamese/), *nukhngcprojectnychcmnhphimtrtnhiuthigianmicOcphinbntexttospeechOutin*.

Tóm lại thì việc sinh ra tiếng nói từ text gồm 4 giai đoạn

1. Sinh ra features từ file wav sử dụng tool sptk 2. Tạo một lab, trong đó có dữ liệu huấn luyện (những đặc trưng của âm thanh được trích xuất từ bước 1), text đầu vào 3. Sử dụng htk để train dữ liệu từ thư mục lab, đầu ra là một model 4. Sử dụng model để sinh ra output với text đầu vào, dùng *hts_engineOdecode, ktquOcwav files*.

Phù. 4 bước đơn giản thế này thôi mà không biết. Lọc cả internet ra mãi chẳng hiểu, cuối cùng file phân tích file 'train.sh' của bạn Truong Do mới hiểu. Ahihi

****24/11/2017****: Nhánh của Trí tuệ nhân tạo mà hiện tại mình đang theo đuổi. Project hiện tại là [underthesea](https://github.com/magizbox/underthesea). Với mục đích là xây dựng một toolkit cho xử lý ngôn ngữ tự nhiên tiếng Việt.

Chương 12

Nhận dạng tiếng nói

Trong hệ thống nhận dạng tiếng nói, tín hiệu âm thanh được thu thập như những mẫu phù hợp cho quá trình xử lý của máy tính và được đưa vào quá trình nhận diện. Đầu ra của hệ thống là một câu phụ đề của câu nói.

Nhận dạng tiếng nói là một nhiệm vụ phức tạp và hệ thống tốt nhất trong nhận dạng tiếng nói rất phức tạp. Có rất nhiều cách tiếp cận cho mỗi thành phần. Trong phần này, người viết chỉ muốn đưa ra một cái nhìn tổng thể về nhận dạng tiếng nói, các khó khăn chính, các thành phần cơ bản, chức năng và tương tác của chúng trong một hệ thống nhận dạng tiếng nói.

Các thành phần của hệ thống nhận dạng tiếng nói

Trong bước thứ nhất, trích rút thông tin *Feature Extraction*, các mẫu tín hiệu được tham số hóa. Mục tiêu là trích xuất ra một tập các tham số (đặc trưng) từ tín hiệu có nhiều thông tin hữu ích nhất cho quá trình phân loại. Các đặc trưng chính được trích xuất với điều kiện *thích nghi* với các sự thay đổi của âm thanh và *nhảy cảm* với các nội dung ngôn ngữ.

Trong module phân loại, các vector đặc trưng được ánh xạ với các pattern, được gọi là *mô hình âm học* (acoustic model). Mô hình học thường là HMM được train với toàn bộ từ, hay âm như là một đơn vị ngôn ngữ.

Từ điển phát âm (pronunciation dictionary) định nghĩa cách kết hợp âm cho các ký tự. Nó có thể chứa cách phát âm khác nhau cho cùng một từ. Bảng 1 hiển thị chính xác một từ điển. Từ (grapheme) ở cột bên trái ứng với cách phát âm (các âm) ở cột bên phải (các ký tự âm trong bảng được dùng phổ biến đối với tiếng Anh)

word pronunciation	INCREASE	ih n
k r iy s	INCREASED	ih n k r iy s t
INCREASES	ih n k r iy s ah z	
INCREASING	ih n k r iy s ih ng	
INCREASINGLY	ih n k r iy s ih ng l iy	
INCRECIBLE	ih n k r eh d ah b ah l	

Mô hình ngôn ngữ (language model) chứa các thông tin về cú pháp. Mục tiêu để dự đoán khả năng một từ xuất hiện sau các từ khác trong một ngôn ngữ. Nói cách khác, xác suất để một từ k xảy ra sau khi $k-1$ từ sau đó được định nghĩa bởi $latexP(w_k|w_{k-1}, w_{k-2}, ..., w_1)$

Mô hình hóa sub-word với HMMs

Trong các hệ thống ASR, HMMs được dùng để biểu diễn các đơn vị dưới từ (ví dụ như âm). Với ngôn ngữ, thông thường có 40 âm. Số lượng âm phụ thuộc

vào từ điển được sử dụng. Số lượng âm phụ thuộc vào từ điển được sử dụng. Mô hình từ có thể được xây dựng bằng cách kết hợp các mô hình dưới từ.

Trong thực tế, khi nhận dạng một âm phụ thuộc rất nhiều vào các âm bên cạnh. Do đó, mô hình âm phụ thuộc ngữ cảnh (*context dependence*) được sử dụng rất phổ biến. Mô hình *biphone* chú ý đến âm bên trái hoặc âm bên phải, mô hình *triphone* chú ý đến cả hai phía, với một âm, các mô hình khác nhau được sử dụng trong ngữ cảnh khác nhau. Hình dưới thể hiện các mô hình monophone, biphone và triphone của từ *bat* (b ae t)

Quá trình huấn luyện

****Huấn luyện các mô hình monophone****

Một mô hình monophone là một mô hình âm học, trong đó không chứa thông tin ngữ cảnh về các âm trước và sau. Nó được sử dụng như thành phần cơ bản cho các mô hình triphone - mô hình sử dụng những thông tin về ngữ cảnh.

Việc huấn luyện sử dụng framework Gaussian Mixture Model/Hidden Markov Model.

****Đóng hàng âm thanh trong mô hình âm học****

Các tham số trong mô hình âm học được tính toán trong quá trình huấn luyện; tuy nhiên, quá trình này có thể được tối ưu hóa bởi việc lặp lại quá trình huấn luyện và đóng hàng. Còn lại là huấn luyện Viterbi (liên quan đến phương pháp này, nhưng dùng nhiều khối lượng tính toán hơn là thuật toán Forward-Backward và Expectation Maximization). Bằng cách đóng hàng âm thanh - phụ đề với mô hình âm học hiện tại, các thuật toán huấn luyện có thể sử dụng kết quả này để cải thiện và hiệu chỉnh tham số của mô hình. Do đó, mỗi quá trình huấn luyện sẽ theo bởi một bước đóng hàng trong đó âm thanh và văn bản được đóng hàng lại.

****Huấn luyện các mô hình triphone****

Trong khi các mô hình monophone đơn giản biểu diễn các đặc trưng âm thanh như một đơn âm, trong khi các âm vị sẽ thay đổi đáng kể phụ thuộc vào ngữ cảnh. Mô hình triphone thể hiện một âm trong ngữ cảnh với hai âm bên cạnh.

Đến đây, một vấn đề là không phải tất cả các đơn vị triphone được thể hiện trong dữ liệu huấn luyện. Có tất cả (of phonemes)³ *triphone, nhngchcmttpthcstntitrongdliu.Hnna, ccQnvxyr*

****Đóng hàng các mô hình âm học và huấn luyện lại các mô hình triphone****

Lặp lại các bước dòng hàng âm thanh và huấn luyện các mô hình triphone với các thuật toán huấn luyện để hiệu chỉnh mô hình. Các phương pháp phổ biến là delta+delta-delta, LDA-MLLT và SAT. Các giải thuật đóng hàng bao gồm đóng hàng cho từng người nói và FMLLR.

****Các thuật toán huấn luyện****

Huấn luyện delta+delta-delta tính các đặc trưng delta và double-delta, hay các hệ số động, để thêm vào các đặc trưng MFCC. Delta và delta-delta là các đặc trưng số học, tính các đạo hàm bậc 1 và 2 của tín hiệu. Do đó, phép tính toán này thường được thực hiện trên một window của các đặc trưng vector. Trong khi một window của hai đặc trưng vector có thể hiệu quả, nó là các xấp xỉ thô (giống như delta-difference là một xấp xỉ thô của đạo hàm). Đặc trưng delta được tính toán trong các window của các đặc trưng cơ bản, trong khi delta-delta được tính toán trong các window của đặc trưng delta.

LDA-MLLT viết tắt của Linear Discriminant Analysis - Maximum Likelihood Linear Transform. Linear Discriminant Analysis lấy các đặc trưng vector

và xây dựng các trạng thái HMM, nhưng giảm thiểu không gian vector. Maximum Likelihood Linear Transform lấy các đặc trưng được giảm từ LDA, và thực hiện các biến đổi đối với từng người nói. MLLT sau đó thực hiện một bước chuẩn hóa, để giảm sự khác biệt giữa các người nói.

SAT viết tắt của Speaker Adaptive Training. SAT cũng thực hiện các chuẩn hóa đối với người nói bằng cách thực hiện biến đổi trên mỗi người nói. Kết quả của quá trình này đồng nhất và chuẩn hóa hơn, cho phép mô hình có thể sử dụng những tham số này để giảm thiểu sự biến đổi của âm, đối với từng người nói hoặc môi trường thu.

****Các thuật toán đóng hàng****

Thuật toán dòng hàng luôn luôn cố định, trong đó các kịch bản chấp nhận các loại đầu vào âm học khác nhau. Dòng hàng đối với từng người nói, sẽ tách biệt thông tin giữa các người nói trong quá trình đóng hàng.

fMLLR viết tắt của Feature Space Maximum Likelihood Linear Regression. Sau quá trình huấn luyện SAT, các mô hình âm học không huấn luyện trên các đặc trưng ban đầu, mà đối với các đặc trưng chuẩn hóa theo người nói. Với quá trình đóng hàng, xóa bỏ sự khác biệt giữa người nói (bằng cách nghịch đảo ma trận fMLLR), sau đó loại bỏ nó khỏi mô hình *bằng cách nhân ma trận nghịch đảo với đặc trưng vector). Mô hình âm học quasi-speaker-independent có thể sử dụng trong quá trình đóng hàng.

Dòng hàng (Forced Alignment)

Hệ thống nhận dạng tiếng nói sử dụng một máy tìm kiếm bên cạnh mô hình âm học và ngôn ngữ trong đó chứa tập các từ, âm và tập dữ liệu để đối chiếu với dữ liệu âm thanh cho câu nói. Máy tìm kiếm này sử dụng các đặc trưng được trích xuất bởi dữ liệu âm thanh để xác định sự xuất hiện của từ, âm và đưa ra kết quả.

Quá trình dòng hàng cũng tương tự như vậy, nhưng khác ở một điểm quan trọng. Thay vì đưa vào tập các từ có thể để tìm kiếm, máy tìm kiếm đưa vào đoạn phụ đề tương ứng với câu nói. Hệ thống sau đó đóng hàng dữ liệu văn bản với dữ liệu âm thanh, xác định đoạn nào trong âm thanh tương ứng với từ cụ thể nào trong dữ liệu văn bản.

Dòng hàng có thể sử dụng để đóng âm trong dữ liệu với bản với dữ liệu âm thanh, giống như hình dưới đây, các âm được xác định trong từng đoạn của âm thanh.

Hidden Markov Model

Hidden Markov Model (HMM) là mô hình trọng số với các trọng số ở cung, chỉ khả năng xuất hiện của cung.

Một trong những ứng dụng của HMM, là phán đoán chuỗi các trạng thái thay đổi, dựa vào chuỗi các quan sát

Các trọng số trong trạng thái gọi là observation likelihood, các trọng số ở cung gọi là transition likelihood.

Sau đây là một ví dụ:

* Thời tiết trong một ngày có thể là NÓNG hoặc LẠNH * Khi trời NÓNG,
20* Khi trời NÓNG, 30* (
qimg-a6744f9e17e59f3729d6fef02d54391b.webp)

Giờ, giả sử chúng ta quan sát trong 3 ngày, bạn dùng 1,2,3 viên đá. Thời tiết có khả năng diễn ra như thế nào?

Đến đây chúng ta dùng thuật toán Viterbi. Về cơ bản, nó là dynamic programming với hai chiều $[state, position_{in_sequence}]$

Gọi S là trạng thái hiện tại HOT, COLD trong quan sát i , S' là trạng thái trước đó, và A là lượng đá tiêu thụ 1, 2, 3 trong quan sát i

$$Viterbi[S, i] = Viterbi[S', i - 1] * p(S|S') * p(A|S)$$

$$V[S, i] = V[S', i - 1] * transition_{ikelihood} * observation_{ikelihood}$$

HMM được sử dụng trong các hệ thống thoại miễn

1. Có hữu hạn các trạng thái nội tại (internal state), là nguyên nhân của các sự kiện (external events) (các quan sát) 2. Trạng thái nội tại không quan sát được (hidden) 3. Trạng thái hiện tại chỉ phụ thuộc vào trạng thái trước đó (quá trình Markov)

Wow! George nhanh chóng liên hệ vụ của anh đây với mô hình HMM. George nhận ra rằng CCTV footage từ các cặp có thể coi như là chuỗi quan sát được, anh đây có thể dùng mô hình và sử dụng nó để phát hiện hành vi ẩn mà Bob và William hoạt động.

****3 vấn đề cơ bản**** được Jack Ferguson giới thiệu trong những năm 1960

Vấn đề 1 (Likelihood): Cho một HMM $\lambda = (A, B)$ và một chuỗi quan sát O , xác định likelihood $P(O|\lambda)$

Vấn đề 2 (Decoding): Cho một chuỗi quan sát O , và một HMM $\lambda = (A, B)$, xác định chuỗi ẩn Q tốt nhất

Vấn đề 3 (Learning): Cho một chuỗi quan sát O , một tập các trạng thái trong HMM, học các tham số A và B

****Likelihood Computation****

Vấn đề đầu tiên là tính xác suất xảy ra của một chuỗi quan sát. Ví dụ, trong bài toán ăn đá ở hình 9.3, xác suất xảy ra chuỗi *3 1 3* là bao nhiêu?

****Tính toán Likelihood****: Chuỗi một HMM $\lambda = (A, B)$, và mỗi chuỗi quan sát O , xác định likelihood $P(O|\lambda)$

Thuật toán Forward, nếu sử dụng Bayes rule, để tính likelihood, cần khối lượng tính toán N^T với N là số trạng thái có thể có và T là chiều dài chuỗi quan sát. Ví dụ trong bài toán gán nhãn có $N=10$ nhãn, chiều dài của chuỗi trung bình là 28, thì cần 10^{28} bước tính toán. Một giải thuật với hiệu quả $O(N^2T)$ được đề xuất với tên gọi ****forward algorithm****

Tài liệu tham khảo

* <http://www.igi.tugraz.at/lehre/CI/SS08/tutorials/ASR/node1.html> * <https://www.isip.piconepress.com/http://www.igi.tugraz.at/lehre/CI/SS08/tutorials/ASR/node1.html> * <https://www.isip.piconepress.com/projects/speech/software/tutorials/production/fundamentals/v1.0/section> * <https://www.quora.com/What-is-a-simple-explanation-of-the-Hidden-Markov-Model-algorithm>

Chương 13

Phân loại văn bản

Naive Bayes Classifier Tham khảo thư viện http://scikit-learn.org/stable/modules/naive_bayes.html *Scikit – learn*

Xét bài toán classification với C classes $1, 2, \dots, C$. Tính xác suất để 1 điểm dữ liệu rơi vào class C ta có công thức: $P(\frac{c}{x})$. Tức tính xác suất để đầu ra là class C biết rằng đầu vào là vector x . Việc xác định class của điểm dữ liệu đó bằng cách chọn ra class có xác suất cao nhất: $c = \operatorname{argmax}(P(\frac{c}{x}))$ với $c = 1, \dots, C$ Sử dụng quy tắc Bayes: $c = \operatorname{argmax}(P(\frac{c}{x})) = \operatorname{argmax}(P(\frac{P(\frac{c}{x})P(x)}{P(x)}) = \operatorname{argmax}(P(\frac{P(\frac{c}{x})}{P(c)}))$

Các phân phối thường dùng **Gaussian Naive Bayes**

Mô hình này được sử dụng chủ yếu trong loại dữ liệu mà các thành phần là các biến liên tục. **Multinomial Naive Bayes** Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà feature vectors được tính bằng Bags of Words. Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển. Giá trị của thành phần thứ i trong mỗi vector chính là số lần từ thứ i xuất hiện trong văn bản đó. Khi đó, $P(\frac{x_i}{c})$ tỉ lệ với tần suất từ thứ i xuất hiện trong các văn bản của class c : $P(\frac{x_i}{c}) = \frac{N_{x_i}}{N_c}$ Trong đó: N_{x_i} là tổng số lần từ thứ i xuất hiện trong các văn bản của class c , nó được tính là tổng của tất cả các thành phần thứ i của các feature vectors ứng với class c . N_c là tổng số từ (kể cả lặp) xuất hiện trong class c . Hay bằng tổng độ dài của toàn bộ các văn bản thuộc vào class c . Nếu có một từ mới chưa bao giờ xuất hiện trong class c thì biểu thức trên sẽ bằng 0, điều này dẫn đến vế phải của c bằng 0. **Bernoulli Naive Bayes** Mô hình này được áp dụng cho các loại dữ liệu mà mỗi thành phần là một giá trị binary. Ví dụ: cũng với loại văn bản nhưng thay vì đếm tổng số lần xuất hiện của 1 từ trong văn bản, ta chỉ cần quan tâm từ đó có xuất hiện hay không. Khi đó: $P(\frac{x_i}{c}) = P(\frac{i}{c})x_i + (1 - P(\frac{i}{c}))(1 - x_i)$ Với $P(\frac{i}{c})$ là xác suất từ thứ i xuất hiện trong các văn bản của class c .

Chương 14

Pytorch

****Bí kíp luyện công****

(cập nhật 08/12/2017): cảm giác [talk](http://videlectures.net/deeplearning2017_chintala_torch/) của anh Soumith Chintala

Sau khi nghe bài này thì hôm mộ luôn anh Soumith Chintala, tìm loạt bài anh trình bày luôn

* [PyTorch: Fast Differentiable Dynamic Graphs in Python with a Tensor JIT](https://www.youtube.com/watch?v=DBVLcgq2Eg0&t=2s), Strange Loop Sep 2017 * [Keynote: PyTorch: Framework for fast, dynamic deep learning and scientific computing](https://www.youtube.com/watch?v=LAMwEJZqesU&t=66s), EuroSciPy Aug 2017

So sánh giữa Tensorflow và Pytorch?

Có 2 điều cần phải nói khi mọi người luôn luôn so sánh giữa Tensorflow và Pytorch. (1) Tensorflow khiến mọi người "không thoải mái" (2) Pytorch thực sự là một đối thủ trên bàn cân. Một trong những câu trả lời hay nhất mình tìm được là của anh Hieu Pham (Google Brain) [trả lời trên quora (25/11/2017)](https://www.quora.com/What-are-your-reviews-between-PyTorch-and-TensorFlow/answer/Hieu-Pham-20?srid=5O2u). Điều quan trọng nhất trong câu trả lời này là **"Dùng Pytorch rất sướng cho nghiên cứu, nhưng scale lên mức business thì Tensorflow là lựa chọn tốt hơn"**

Behind The Scene

(15/11/2017) Hôm nay bắt đầu thử nghiệm pytorch với project thần thánh classification sử dụng cnn <https://github.com/Shawn1993/cnn-text-classification-pytorch>

Cảm giác đầu tiên là make it run khá đơn giản

“conda create -n test-torch python=3.5 pip install http://download.pytorch.org/whl/cu80/torch-0.2.0.post3-cp35-cp35m-manylinux1_x86_64.whl pip install torchvision pip install torchtext”

Thế là ‘main.py’ chạy! Hay thật. Còn phải vọc để bạn này chạy với CUDA nữa.

****Cài đặt CUDA trong ubuntu 16.04****

Kiểm tra VGA

“lspci|grepVGA01 : 00.0VGAcompatiblecontroller : NVIDIA Corporation GM204 [GeForce GTX 980] (rev 10)”

Kiểm tra CUDA đã cài đặt trong Ubuntu [1]

“nvcc --versionnvcc : NVIDIA(R)Cuda compiler driver Copyright (c) 2005–

2016 NVIDIA Corporation, Builton Sun Sep 4 2:14:01 CDT 2016 Cuda compilation tools, release 8.0, V8.0.44”

Kiểm tra pytorch chạy với cuda ‘test_cuda.py’

“python import torch print("Cuda:", torch.cuda.is_available())”

“`pythontest_cuda.pyCUDA : True`”

Chỉ cần cài đặt thành công CUDA là pytorch tự work luôn. Ngon thật!

Ngày X

Chẳng hiểu sao update system kiểu nào mà hôm nay lại không sử dụng được CUDA ‘`torch.cuda.is_available() = False`’. *Sau khi dùng `torch.Tensor().cuda()` thì gpl*

“AssertionError: The NVIDIA driver on your system is too old (found version 8000). Please update your GPU driver by downloading and installing a new version from the URL: <http://www.nvidia.com/Download/index.aspx> Alternatively, go to: <https://pytorch.org/binaries> to install a PyTorch version that has been compiled with your version of the CUDA driver.”

Kiểm tra lại thì mình đang dùng nvidia-361, làm thử theo [link này](<http://www.linuxandubuntu.com/home/to-install-latest-nvidia-drivers-in-linux>) để update NVIDIA, chưa biết kết quả ra sao?

May quá, sau khi update lên nvidia-387 là ok. Haha

Ngày 2

Hôm qua đã bắt đầu implement một nn với pytorch rồi. Hướng dẫn ở [Deep Learning with PyTorch: A 60 Minute Blitz]([http://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.h](http://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html)

Hướng dẫn implement các mạng neural với pytorch rất hay tại [PyTorch-Tutorial](<https://github.com/MorvanZhou/PyTorch-Tutorial>)

(lướt lướt) Trang này [Awesome-pytorch-list](<https://github.com/bharathgs/Awesome-pytorch-list>) chứa rất nhiều link hay về pytorch như tập hợp các thư viện liên quan, các hướng dẫn và ví dụ sau đó là các cài đặt của các paper sử dụng pytorch.

(lướt lướt) Loạt video hướng dẫn pytorch [PyTorchZeroToAll](<https://www.youtube.com/watch?v=SKq-pmkekTkamp;list=PLlMkM4tgfjnJ3I-dbhO9JT7gNty6o2m>) *catcgiSungKimtrnyoutube*.

Bước tiếp theo là visualize loss và graph trong tensorboard, sử dụng [tensorboard_logger](https://github.com/TeamHG-Memex/tensorboard_logger) *khay*.

“`pip install tensorboard_loggerpipinstalltensorboard`”

Chạy tensorboard server

“`tensorboard --log-dir=runs`”

Ngày 3: Vấn đề kỹ thuật

Hôm qua cố gắng implement một phần thuật toán CNN cho bài toán phân lớp văn bản. Vấn đề đầu tiên là biểu diễn sentence thế nào. Cảm giác load word vector vào khá chậm. Mà thằng tách từ của underthesea cũng chậm kinh khủng.

Một vài link tham khảo về bài toán CNN: [Implementing a CNN for Text Classification in TensorFlow](<http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/>), [Text classification using CNN : Example](<https://agarnitin86.github.io/text-classification-cnn/>)

[¹] : <https://askubuntu.com/questions/799184/how-can-i-install-cuda-on-ubuntu-16-04>

Phần V

Linh tinh

Chương 15

Nghiên cứu

Các công cụ

[Google Scholar](https://scholar.google.com.vn/) vẫn là lựa chọn tốt

* Tìm kiếm tác giả theo lĩnh vực nghiên cứu và quốc gia: sử dụng filter label: + đuôi * ví dụ: [danh sách các nhà nghiên cứu Việt Nam thuộc lĩnh vực xử lý ngôn ngữ tự nhiên (label:natural_language_processing + .vn)](https://scholar.google.com.vn/citations?hl=en&view_op=search_authors&mauthors=label*danh_schny₀sxptheolngtrchdn

Bên cạnh đó còn có [semanticscholar](https://www.semanticscholar.org/) (một project của [allenai](http://allenai.org/)) với các killer features

* [Tìm kiếm các bài báo khoa học với từ khóa và filter theo năm, tên hội nghị](https://www.semanticscholar.org/search?venue*) [Xem những người ảnh hưởng, ảnh hưởng bởi một nhà nghiên cứu, cũng như xem co-author, journals và conferences mà một nhà nghiên cứu hay gửi bài](https://www.semanticscholar.org/author/Christopher-D-Manning/1812612)

Mendeley rất tốt cho việc quản lý và lưu trữ. Tuy nhiên điểm hạn chế lại là không lưu thông tin về citation

Các hội nghị tốt về xử lý ngôn ngữ tự nhiên

* Rank A: ACL, EACL, NAACL, EMNLP, CoNLL * Rank B: SemEval

Các tạp chí

* [Computational Linguistics (CL)](http://www.mitpressjournals.org/loi/coli)

Câu chuyện của Scihub

Sci-Hub được tạo ra vào ngày 5 tháng 9 năm 2011, do nhà nghiên cứu đến từ Kazakhstan, [Alexandra Elbakyan](https://en.wikipedia.org/wiki/Alexandra_Elbakyan)

Hãy nghe chia sẻ của cô về sự ra đời của Sci-Hub

> Khi tôi còn là một sinh viên tại Đại học Kazakhstan, tôi không có quyền truy cập vào bất kỳ tài liệu nghiên cứu. Những bài báo tôi cần cho dự án nghiên cứu của tôi. Thanh toán 32 USD thì thật là điên rồ khi bạn cần phải đọc lướt hoặc đọc hàng chục hoặc hàng trăm tờ để làm nghiên cứu. Tôi có được những bài báo như vậy vào trộm chúng. Sau đó tôi thấy có rất nhiều và rất nhiều nhà nghiên cứu (thậm chí không phải sinh viên, nhưng các nhà nghiên cứu trường đại học) giống như tôi, đặc biệt là ở các nước đang phát triển. Họ đã tạo ra các cộng đồng trực tuyến (diễn đàn) để giải quyết vấn đề này. Tôi là một thành viên tích cực trong một cộng đồng như vậy ở Nga. Ở đây ai cần có một bài nghiên cứu, nhưng không thể trả tiền cho nó, có thể đặt một yêu cầu và các thành viên

Về phần mình, là một nhà nghiên cứu trẻ, đương nhiên phải đọc liên tục. Các báo cáo ở Việt Nam về xử lý ngôn ngữ tự nhiên thì thường không tải lên các trang mở như arxiv.org, các kỷ yếu hội nghị cũng không public các proceedings. Thật sự scihub đã giúp mình rất nhiều.

Đành phải cài tor để truy cập vào scihub ở địa chỉ <http://scihub222660qcxt.onion/> <https://dl.acm.org/citation.cfm?id=3242818>
Và mọi chuyện lại ổn.

* Làm việc mỗi ngày * Viết nhật ký nghiên cứu mỗi tuần (tổng kết công việc tuần trước, các ý tưởng mới, kế hoạch tuần này) * Cập nhật các kết quả từ các hội nghị, tạp chí

* [Machine Learning Yearning, by Andrew Ng] (<https://gallery.mailchimp.com/dc3a7ef4d750c0abfc19202a3>)

* Review các khóa học Deep Learning: <https://www.kdnuggets.com/2017/10/3-popular-courses-deep-learning.html>

(01/11/2017) Không biết mình có phải làm nghiên cứu không nữa? Vừa
kiêm phát triển, vừa đọc paper mỗi ngày. Thôi, cứ (miễn cưỡng) cho là nghiên
cứu viên đi.

Chương 16

Nghề lập trình

Chân kinh con đường lập trình: [Teach Yourself Programming in Ten Years. Peter Norvig](<http://norvig.com/21-days.html>)

Trang web hữu ích

* Chia sẻ thú vị: [15 năm lập trình ở Việt Nam](<https://vozforums.com/showthread.php?t=3431312>) của Blanic (vozfourm) * Trang web chứa cheatsheet so sánh các ngôn ngữ lập trình và công nghệ <http://hyperpolyglot.org/>

01/11/2017

Vậy là đã vào nghề (đi làm full time trả lương) được 3 năm rưỡi rồi. Thời gian trôi qua nhanh như *ó chạy ngoài đồng thật. Tâm đắc nhất với câu trong một quyển gì đó của anh lead HR google. Có 4 level của nghề nghiệp. 1 là thỏa mãn được yêu cầu cả bản. 2 là dự đoán được tương lai. 3 là cá nhân hóa (ý nói là tận tình với các khách hàng). 4 là phiêu diêu tự tại. Hay thật! Bao giờ mới được vào mức 4 đây.

Chương 17

Latex

15/12/2017:

Hôm nay tự nhiên nổi hứng vẽ hình trên latex. Thấy blog này là một guide line khá tốt về viết blog phần mềm. Quyết định cài latex

Theo [hướng dẫn này](<http://milq.github.io/install-latex-ubuntu-debian/>)

“ sudo apt-get install texlive-full sudo apt-get install texmaker “

Tìm được ngay bên này <https://www.overleaf.com/> có vẻ rất hay luôn

Hướng dẫn cực kì cơ bản <http://www.math.uni-leipzig.de/hellmund/LaTeX/pgf-tut.pdf>

Chương trình đầu tiên, vẽ diagram cho LanguageFlow

```
\documentclass[border=10pt]{standalone}
\usepackage{verbatim}
\begin{comment}
\end{comment}
\usepackage{tikz}
\begin{document}
\begin{tikzpicture}
  \node[draw] (model) at (0, 0) {Model Folder};
  \node[draw] (analyze) at (6, 0) {Analyze Folder};
  \node[draw] (board) at (3,2) {Board};
  \node[draw] (logger) at (3, -2) {Logger};

  \path[->, densely dotted] (board.east)
    edge [out=0, in=90]
    node[fill=white, pos=.5] {\tiny (1) init}
    (analyze.north) ;
  \path[->, densely dotted] (board.south)
    edge [out=-90, in=180]
    node[fill=white, pos=.3] {\tiny (2) serve}
    (analyze.west) ;
  \path[->, densely dotted] (logger.west)
    edge [out=180, in=-90]
    node[fill=white, pos=.7] {\tiny (1) read}
    (model.south) ;
  \path[->, densely dotted] (logger.east)
```

```
edge [out=0, in=-90]
node[fill=white, pos=.7] {\tiny (2) write}
(analyze.south) ;
\end{tikzpicture}
\end{document}
```

Doc! Doc! Doc! <https://en.wikibooks.org/wiki/LaTeX/PGF/TikZ>

Chương 18

Chào hàng

****16/01/2018**** Bối cảnh. Hôm nay gửi lời mời kết bạn đến một thằng làm research về speech mà nó "chửi" mình không biết pitch. Tổ sư. Tuy nhiên, nó cũng dạy mình một bài học hay về pitch.

Chửi nó là vậy nhưng lần sau sẽ phải đầu tư nhiều hơn cho các lời pitch.

Vẫn không ưa Huyền Chíp như ngày nào, nhưng [bài này](<https://www.facebook.com/notes/huyen-chip/k>)

Tóm lại skill này có 4 phần

1. Ngôn ngữ không trau chuốt 2. Giới thiệu bản thân không tốt 3. Không chỉ ra cho người nhận rằng họ sẽ được gì 4. Không có phương án hành động

Đối với email, thì cần triển khai thể này

* [Chào hỏi] * [Giới thiệu bản thân một cách nào đó để người đọc quan tâm đến bạn] * [Giải thích lý do bạn biết đến người này và bạn ấn tượng thế nào với họ – ai cũng thích được nghe khen] * [Bạn muốn gì từ người đó và họ sẽ được gì từ việc này] * [Kết thúc]

Chương 19

Phát triển phần mềm

* Phát triển phần mềm là một việc đau khổ. Từ việc quản lý code và version, packing, documentation. Dưới đây là lược lặt những nguyên tắc cơ bản của mình.

Quản lý phiên bản

Việc đánh số phiên bản các thay đổi của phần mềm khi có hàm được thêm, lỗi được sửa, hay các phiên bản tiền phát hành cần thống nhất theo chuẩn của [semversion]. Điều này giúp nhóm có thể tương tác dễ hơn với người dùng cuối.

)

****Đánh số phiên bản****

Phiên bản được đánh theo chuẩn của [semversion](<https://semver.org/>).

* Mỗi khi một bug được sửa, phiên bản sẽ tăng lên một patch. * Mỗi khi có một hàm mới được thêm, phiên bản sẽ tăng lên một patch. * Khi một phiên bản mới được phát hành, phiên bản sẽ tăng lên một minor. * Trước khi phát hành, bắt đầu với x.y.z-rc, x.y.z-rc.1, x.y.z-rc.2. Cuối cùng mới là x.y.z * Mỗi khi phiên bản rc lỗi, khi public lại, đặt phiên bản alpha x.y.z-alpha.t (một phương án tốt hơn là cài đặt thông qua github)

****Đánh số phiên bản trên git****

Ở nhánh develop, mỗi lần merge sẽ được đánh version theo PATCH, thể hiện một bug được sửa hoặc một thay đổi của hàm

Ở nhánh master, mỗi lần release sẽ được thêm các chỉ như x.y1.0-rc, x.y1.0-rc.1, x.y1.0-rc, x.y1.0

Vẫn còn lẩn tẩn:

* Hiện tại theo workflow này thì chưa cần sử dụng alpha, beta (chắc là khi đó đã có lượt người sử dụng mới cần đến những phiên bản như thế này)

****Tải phần mềm lên pypi****

Làm theo hướng dẫn [tại đây](<http://peterdowns.com/posts/first-time-with-pypi.html>)

1. Cấu hình file ‘.pypirc’ 2. Upload lên pypi

“ python setup.py sdist upload -r pypi “

Chương 20

Phương pháp làm việc

Xây dựng phương pháp làm việc là một điều không đơn giản. Với kinh nghiệm 3 năm làm việc, trải qua 2 project. Mà vẫn chưa produce được sản phẩm cho khách hàng. Thiết nghĩ mình nên viết phương pháp làm việc ra để xem xét lại. Có lẽ sẽ có ích cho mọi người.

Làm sao để làm việc hiệu quả, hay xây dựng phương pháp làm việc hữu ích? Câu trả lời ngắn gọn là "Một công cụ không bao giờ đủ".

<!-more->

Nội dung

1. [Làm sao để đánh giá công việc trong khoảng thời gian dài hạn?](section1)
2. [Làm sao để quản lý project?](section2)
3. [Làm sao để công việc trôi chảy?](section3)
4. [Làm sao để xem xét lại quá trình làm việc?](section4)

<p id="section1">nbsp;</p>

Làm sao để đánh giá công việc trong khoảng thời gian dài hạn?

Câu trả lời OKR (Objectives and Key Results)

 OKR Framework

Đầu mỗi quý , nên dành vài ngày cho việc xây dựng mục tiêu và những kết quả quan trọng cho quý tới. Cũng như review lại kết quả quý trước.

Bước 1: Xây dựng mục tiêu cá nhân (Objectives)

Bước 2: Xây dựng các Key Results cho mục tiêu này

Bước 3: Lên kế hoạch để hiện thực hóa các Key Results

<p id="section2">nbsp;</p>

Làm sao để quản lý một project

Meistertask

 * MeisterTask*

<p id="section3">nbsp;</p>

Làm sao để công việc trôi chảy?

Có vẻ trello là công cụ thích hợp

Bước 1: Tạo một team với một cái tên thật ấn tượng (của mình là Strong Coder)

Trong phần Description của team, nên viết Objectives and Key Results của quý này

Sau đây là một ví dụ

“ Objectives and Key Results

-> Build Vietnamese Sentiment Analysis -> Develop underthesea -> Deep Learning Book “

Bước 2: Đầu mỗi tuần, tạo một board với tên là thời gian ứng với tuần đó (của mình là ‘2017 | Fight 02 (11/12 - 16/12)‘)

Board này sẽ gồm 5 mục: "TODO", "PROGRESSING", "Early Fight", "Late Fight", "HABBIT", được lấy cảm hứng từ Kanban Board

)
TrelloBoardexample*

* Mỗi khi không có việc gì làm, xem xét card trong "TODO" * [FOCUS] tập trung làm việc trong "PROGRESSING" * Xem xét lại thói quen làm việc với "HABBIT"

Một Card cho Trello cần có

* Tên công việc (title) * Độ quan trọng (thể hiện ở label xanh (chưa quan trọng), vàng (bình thường), đỏ (quan trọng)) * Hạn chót của công việc (due date)

Sắp xếp TODO theo thứ tự độ quan trọng và Due date

<p id="section4">nbsp;</p>

Làm sao để xem xét lại quá trình làm việc?

Nhật lý làm việc hàng tuần . Việc này lên được thực hiện vào đầu tuần . Có 3 nội dung quan trọng trong nhật ký làm việc (ngoài gió mây trăng cảm xúc, quan hệ với đồng nghiệp...)

* Kết quả công việc tuần này * Những công việc chưa làm? Lý do tại sao chưa hoàn thành? * Dự định cho tuần tới

Đang nghiên cứu

Làm sao để lưu lại các ý tưởng, công việc cần làm?: Dùng chức năng checklist của card trong meister. Khi có ý tưởng mới, sẽ thêm một mục trong checklist

Làm sao để tập trung vào công việc quan trọng?: Dùng chức năng tag của meister, mỗi một công việc sẽ được đánh sao (với các mức 5 sao, 3 sao, 1 sao), thể hiện mức độ quan trọng của công việc. Mỗi một sprint nên chỉ tập trung vào 10 star, một product backlog chỉ nên có 30 star.

Tài liệu của dự án: Sử dụng Google Drive, tài liệu mô tả dự án sẽ được link vào card tương ứng trong meister.

Tài liệu tham khảo

Goodfellow, Ian / Bengio, Yoshua / Courville, Aaron (2016): *Deep Learning*. , MIT Press.

Hai, Do (2018): *Một số phân phối phổ biến* .

Chỉ mục

convolution, 43