Análisis de Clusters Aplicado a Repositorios Públicos de Github

R. Leslie, 10-10613 Universidad Simón Bolívar
S. Erick, 11-10969 Universidad Simón Bolívar
T. Georvic, 12-11402 Universidad Simón Bolívar

Resumen

Se usaron algorimos de agrupamiento como K-Means, EM, Jerárquico aglomeramiento y Jerárquico divisivo para particionar los lenguajes de programación de un conjunto de datos obtenidos de GitHub. También se calcula una matriz de correlaciones entre los lenguajes. Con estas técnicas se confirmaron agrupamientos esperados como el de programación web y se encontraron otros no vistos previamente, como el de lenguajes con características funcionales que son usados para reemplazar a JavaScript.

1. Introducción

1. Planteamiento del Problema

Los empleadores en el área de las ciencias de la computación necesitan determinar de manera rápida quiénes, entre todos sus candidatos, cumplen de mejor manera con su perfil profesional buscado. Una razón por las cual esto puede ser útil es la reducción de costes por entrevistas de trabajo innecesarias. Ya cuando se hayan identificado a los potenciales empleados que cumplan con un perfil, o se haya filtrado la lista de postulantes, se puede realizar la entrevista.

2. Trabajos relacionados

Este proyecto está basado en un trabajo de investigación hecho por Guevara y Amirali en [1]. A diferencia de éste, su trabajo usa un conjunto de datos masivo que fue recolectado durante varios meses con ayuda de un sistema distribuido con nodos que fungían como *crawlers* y un servidor principal que registraba el progreso de cada uno. En el caso de este proyecto, fueron usados conjuntos de datos desde 8127 repositorios hasta 38558 repositorios.

Adicionalmente, en éste sólo fue necesario usar un *web scraper* para recolectar los datos. Los autores del paper mencionan que el servidor principal de su sistema indexa el último usuario del cual se ha extraído información, para cada nodo. En comparación, aquí sólo se ha programado de manera robusta el *web scraper* de manera tal que al llegar a un límite de peticiones, se almacena el último usuario procesado exitosamente y se anexan los nuevos datos recolectados.

Al igual que en [1], en este proyecto se calculan las correlaciones entre los lenguajes y también se usa el algoritmo K-Means. Sin embargo, la manera en que se calculan los vectores de atributos aquí difiere de [1]. Adicionalmente, en este trabajo se usan diversos algoritmos de clustering como DBSCAN, K-Means, EM, Clustering Jerárquico Divisivo y Clustering Jerárquico Aglomerativo. En el presente trabajo sólo se reportan en detalle los resultados significativos y se mencionan los que no contribuyeron con la resolución del problema. Sin embargo, todos los resultados están adjuntos a este informe.

3. Descripción

Se llevan a cabo tres análisis de clusters en una muestra de repositorios tomados de Github. En el primero, todas las instancias consideradas son los lenguajes encontrados. En el segundo, las instancias son los repositorios. En ambos casos, los atributos son todos los lenguajes encontrados e indican en cada caso el grado de ocurrencias conjuntas con la instancia y buscan determinar relaciones entre ellos. El tercer análisis usa una matriz de similitudes pre calculada y busca determinar qué tan probable es que un usuario use un lenguaje sabiendo que éste usa otros.

4. Objetivo

Determinar si los *clusters* de lenguajes de programación usados de manera conjunta en repositorios de Github revelan los perfiles de programación de los usuarios.

2. Marco Teórico

Aprendizaje no supervisado

Para este proyecto fueron usados algoritmos de aprendizaje no supervisado. Esto se debe a la carencia de un etiquetamiento previo al cómputo de los agrupamientos. Ello implica que en lugar de usar métricas tradicionales para evaluar los *clusters*, se usó una metodología más cualitativa, pues se tienen que encontrar explicaciones a los agrupamientos encontrados. ([3], p. 281)

Agrupamiento

En general, la gran variedad de técnicas de agrupamiento se pueden caracterizar por intentar encontrar una partición de las instancias dadas tal que cada clase contenga puntos que sean lo más parecidos que sea posible. ([2], p. 2)

Correlación de Pearson

Es una medida de la correlación lineal entre dos variables, va de 1 a -1, un valor de 1 representa una correlación lineal positiva, un valor de -1 una correlación lineal negativa y un valor de 0 que no existe correlación lineal entre ambas variables. ([4], pp. 76-77)

3. Diseño de la Solución

1. Algoritmos

EM

Para el primer experimento fue usado el algoritmo de *Expectation Maximization* de Weka, o EM por sus siglas en Inglés, con un número de clusters igual a tres, diez y veinte. Este algoritmo busca maximizar el logaritmo de la probabilidad condicional de que se tengan ciertos clusters dados con una distribución de probabilidad propia dependiendo de los datos presentes. En este caso, se usó una distribución normal. ([2], p. 69)

DBSCAN

Esta técnica requiere dos criterios de entrada para determinar cuáles instancias son *core*, cuáles son *border* y cuáles son ruido. El primer criterio es la distancia máxima dentro de la cual se deben contar las instancias más cercanas. El otro criterio es la cantidad mínima de instancias más cercanas según el primer criterio. Si una instancia cumple con estos dos criterios, entonces se considera *core*. Si no se considera *core*, a pesar de ser cercano a una instancia *core* según el primer criterio, entonces es *border.* Si no es *border* ni *core*, entonces es ruido. ([2], p. 113)

K-Means

Este algoritmo requiere como entrada el número de clusters a computar. El representante de cada cluster es computado a partir de la data. Se inicia con un conjunto inicial de representantes. Luego se obtienen los puntos que pertenecen a cada cluster y se recalcula cada representante usando la media de los clusters. Estos dos pasos se repiten hasta que se alcanza un umbral de cambio. ([2], p. 89)

Jerárquico Aglomerativo

Este usa un método *bottom-up* para ir agrupando cada instancia en clusters cada vez más grandes. Para aglomerar dos clusters distintos existen varios criterios a usar para determinar la distancia entre ellos. En este trabajo se usó la distancia promedio entre sus puntos. Al terminar, el aglomeramiento se puede visualizar como un árbol jerárquico. ([2], p. 103)

Jerárquico Divisivo

Similar al anterior, pero usa un método *top-down*. Se va dividiendo el conjunto de datos mientras se construye un árbol que permite ir categorizando los datos. ([2], p. 104)

2. Recolección y Limpieza de Datos

En primer lugar, se tomó el grafo de seguidores de usuarios, a partir de un usuario raíz dado, y se recorrió en BFS hasta una profundidad específica. Estos usuarios se guardaron en una lista. Posteriormente, para cada repositorio de cada usuario de la lista guardada, se le solicitó a Github información sobre los lenguajes usados en el mismo. Todo este procedimiento se hizo con un *web scraper*. Estos datos se almacenaron en formato JSON. Con esto se generaron tres conjuntos de datos. El primero tiene como usuario raíz a *donnemartin* y suma 8127 repositorios de 491 usuarios y 200 lenguajes. El segundo también tiene como raíz a dicho usuario, pero cuenta con 25898 repositorios y 254 lenguajes. El tercero es la combinación del segundo con la información de los repositorios de otros usuarios encontrados partiendo de la cuenta raíz *GrahamCampbell*. Esto se hizo para generar variedad en el conjunto de datos del último experimento, pues el usuario *donnermartin* se dedica, en general, a la programación de sistemas, mientras que *GrahamCampbell* se dedica a la programación web. Se presume que sus seguidores mantienen intereses similares a estos.

Después de esto, se consideraron distintos tipos de vectores de atributos. Uno de estos consideraba a cada lenguaje de programación como una instancia cuyas dimensiones son, de igual manera, todos los lenguajes. Es decir, para dos lenguajes A y B, se tiene que instancias[“A”][“B”] = N, donde N es el número de veces en que dichos lenguajes son usados juntos. Otra variación de este esquema considera a cada repositorio como una instancia cuyas dimensiones son todos los lenguajes encontrados. En este caso, si j representa a un repositorio y B a un lenguaje, se tiene que instancias[j][“B”] = N, donde N es el número de bytes de dicho lenguaje de programación B en el repositorio j.

Finalmente, en lugar de considerar vectores de atributos, se computaron directamente las similitudes entre los distintos lenguajes calculando las probabilidades condicionales entre ellos. Para llevar a cabo esto, se calculó directamente la matriz de probabilidades condicionales entre cada par de lenguajes atendiendo al número de usuarios que han usado ambos. Es decir, para dos lenguajes A y B, se calcula la probabilidad condicional P(A | B) = P(A y B) / P(B), donde P(B) = (número de usuario que usan B / número de usuarios totales) y P(A y B) = (número de usuarios que usan A y B / número de usuarios totales).

Lenguaje	frecuencia
JavaScript	2118
HTML	1655
CSS	1623
Python	1457
Shell	1252
Ruby	628
Makefile	625

Lenguaje	frecuencia
C++	592
C	590
Objective-C	377
PHP	372
Batchfile	217
TeX	197
CoffeeScript	187

Lenguaje	frecuencia
Perl	183
C#	157
Jupyter Notebook	149
Go	139
Swift	116
Scala	111

Tabla N° 0: Número de repositorios en los que aparece cada lenguaje con más de 100 ocurrencias en todo el conjunto de datos N° 1. Este conjunto tiene 200 lenguajes, 8127 repositorios y 491 usuarios.

3. Implementación

Se usaron dos plataformas distintas: un ecosistema de librerías de minería de datos de Python y la plataforma conocida como Weka, desarrollada en Java.

En el primer caso, se usaron las librerías *scikit-learn* para generar los clusters con los algoritmos DBSCAN, K-Means. Jerárquico Divisivo y Aglomerativo. Pandas fue usado por su fácil integración con scikit-learn durante el proceso de preprocesamiento de datos, el cual sólo implicó normalizar los atributos. Matplotlib fue usado para generar los gráficos del algoritmo jerárquico divisivo. Además de estas librerías, fue implementado un web scraper en Python para recolectar información sobre los repositorios del conjunto de datos. Cuando se usan los scripts implementados con el algoritmo de clustering elegido, se crea un archivo con la asignación de cada instancia a su cluster.

En el segundo caso, se aplicaron en este orden los filtros numericToNominal, Normalize y addCluster de Weka. El primero simplemente le indica que la columna que contiene los nombres de los lenguajes es un atributo nominal. El segundo permite que se normalicen todos los atributos numéricos. El tercero crea un nuevo atributo llamado *cluster* que contiene un atributo nominal con la asignación del cluster por cada instancia. Este último filtro se puede configurar para trabajar con varios algoritmos de agrupamiento, entre los cuales están EM y K-Means. Estos fueron los algoritmo usados en esta plataforma.

4. Experimentos y Resultados

1. Descripción del Primer Experimento

Primera serie de corridas con 8127 repositorios y 491 usuarios. Se encontraron alrededor de 200 lenguajes, entre ellos lenguajes de markup como HTML, lenguajes de programación como C,C++ o Java, y algunas herramientas usadas que se toman como su propio pseudo-lenguaje como yacc, que es un parser de C.

A pesar de que se cuenta con conjuntos de datos con una cantidad considerablemente mayor de repositorios, como en este experimento las instancias son los lenguajes encontrados, su contribución es mínima. De hecho, al aumentar la cantidad de lenguajes de muy poca frecuencia, es posible que la calidad de los clusters disminuya. Esto se puede constatar en el directorio de corridas para EM provisto junto a este informe. En este se registran los resultados para 254 lenguajes 25898 repositorios así como para 265 lenguajes y 38558 repositorios.

2. Resultados del Primer Experimento

En la tabla N° 1 encontramos el agrupamiento hecho por el algoritmo EM con tres clusters. Con colores se indican los *cluster* que se esperan dados los usos de los lenguajes. Es decir, a medida que se va aumentando el número de grupos con este algoritmo, se espera que los nuevos clusters discriminen a los lenguajes marcados con color.

Cluster1	Cluster2	Cluster3
Verde oscuro: Android Negro: web y web-framework Azul: Sistemas y Cómputo Otros: no reconocidos	Magenta oscuro: funcional Púrpura: web con características funcionales y compila a JavaScript. Verde Claro: sistema de manera amigable Cian: circuitos Rojo oscuro: estadística Amarillo oscuro: adobe	Azul: demostraciones automáticas Verde: declarativo o manejo intensivo de datos Rojo claro: diseño de lenguajes
PHP, **Perl**, **Java**, **CSS**, **HTML**, C++, Batchfile, Groff, Fortran, Processing, CMake, **Kotlin**, **FreeMarker**, **Python**, **Ruby**, Makefile, **JavaScript**, Objective-C++, Objective-C, NSIS, Assembly, **Shell**, C, **Handlebars**	**Lua**, **Dart**, **LiveScript**, **Haxe**, **Clojure**, **JSONiq** *, **Elm**, **Rust**, **ColdFusion**, Go, Vala, **D**, Cucumber, Cirru, **Scala**, **Verilog**, Liquid, GLSL, LSL, PowerShell, TeX, **VHDL**, Lean, **Haskell**, **R**, Mask, ABAP, Swift, OpenSCAD, Forth, **Elixir**, Groovy, **OCaml**, **Scheme**, C#, **TypeScript**, Io, Tcl, Nix,	'Graphviz (DOT)', QML, GDScript, Smali, GDB, 'OpenEdge ABL', **Coq**, Rouge, Max, Thrift, ASP, **PLSQL**, 'DIGITAL Command Language', Vue, **Lex**, ApacheConf, KiCad, 'Pure Data', PureBasic, Myghty, SaltStack, Nu, Clarion, 'Game Maker Language', Alloy, Awk,

	XQuery, 'Protocol Buffer', **Matlab**, **Erlang**, **ActionScript**, COBOL, Pascal, Ada, AutoHotkey, '**Common Lisp**', **Julia**, 'Visual Basic', **CoffeeScript**, Eiffel,	PicoLisp, QMake, Eagle, Roff, DTrace, Terra, 'GCC Machine Description', Brainfuck, SystemVerilog, Clean, 'Jupyter Notebook', 'Standard ML', PostScript, HLSL, GAP, 'Module Management System', M4, Harbour, **Idris**, Mako, RenderScript, Racket, XC, Parrot, IDL, Smalltalk, Slash, nesC, Genshi, Perl6, NetLogo, 'Component Pascal', SuperCollider, MAXScript, **SQLPL**, SourcePawn, AspectJ, SAS, 'Vim script', ANTLR, F#, 'Inno Setup', Bison, Gnuplot, 'Emacs Lisp', Smarty, Hy, LLVM, Mercury, **PLpgSQL**, NewLisp, MoonScript, Befunge, Nemerle, UnrealScript, AppleScript, Xtend, Fancy, **Nginx**, FLUX, Arduino, Rebol, UrWeb, Scilab, PureScript, Stan, **Agda,** Stata, HCL, 'POV-Ray SDL', Jasmin, 'AGS Script', Self, PigLatin, XSLT, CLIPS, **Yacc**, Brightscript, M, Prolog, E, XS, Logos, Bro, Nim, XML, Hack, 'Gettext Catalog', Cuda, 'Ragel in Ruby Host', Mathematica, Crystal, Cycript, **Puppet**, EmberScript, VimL,
	Notas: JSONiq es funcional y es comúnmente usado con JavaScript. Elm también es funcional y compila a JavaScript.	

Tabla N° 1: clusters generados para K=3 con EM aplicado al conjunto de datos N° 1

En el cluster N° 1 de la tabla N° 1 encontramos principalmente a dos tipos de lenguajes: aquéllos usados en programación web, tanto del lado del cliente (HTML, CSS y JavaScript) como

del lado del servidor (Perl, PHP, Python y Handlebars), y los lenguajes usados en programación de sistemas de bajo nivel, como C++, C, Makefiles, Assembly, entre otros.

En el cluster N° 2 resaltan los lenguajes de paradigma funcional, así como los simuladores de circuitos, los de cómputo científico, los lenguajes de scripting de Adobe, los de programación segura y amigable de sistemas (Lua, Rust y D) y los lenguajes compilables a JavaScript. De este cluster, resaltan especialmente los últimos, pues algunos de ellos se pueden considerar funcionales. Sin embargo, lo que resalta a este grupo (Dart, TypeScript, Elm, Haxe, CoffeScript y LiveScript) a priori es que surgen con la idea de reemplazar a JavaScript mientras se adoptan nuevas técnicas de lenguajes de programación más recientes. Esto significa que hay una diferencia significativa de uso con respecto a los demás.

Cluster1	Magenta oscuro: funcional Rojo: cómputo científico Verde claro: programación amigable de sistemas	**PHP**, **Lua**, **Clojure**, **D**, **Scala**, TeX, **Haskell**, **R**, **Ruby**, **Elixir**, **OCaml**, **Scheme**, C#, **Matlab**, **Erlang**, '**Common Lisp**'
Cluster2	Rojo claro: diseño de lenguajes Marrón: multimedia	GDScript, GDB, 'DIGITAL Command Language', **Lex**, '**Pure Data**', Awk, PicoLisp, Groff, DTrace, Terra, GAP, M4, Kotlin, Perl6, **SuperCollider**, 'Emacs Lisp', Rebol, **Yacc**, E, Mathematica
Cluster3	Azul: Sistemas	Perl, Java, **C++**, Batchfile, **Makefile, Objective-C, Assembly, C**
Cluster4	Púrpura: web con características funcionales y compilan a JavaScript. Amarillo claro: Adobe Cian: circuitos	**Dart, LiveScript, Haxe**, **JSONiq**, **Elm**, Rust, **ColdFusion**, Go, Vala, Cucumber, Cirru, **Verilog**, Liquid, GLSL, LSL, PowerShell, **VHDL**, Lean, Mask, ABAP, OpenSCAD, Forth, Groovy, **TypeScript**, Io, Tcl, Nix, XQuery, 'Protocol Buffer', **ActionScript**, COBOL, Pascal, Ada, AutoHotkey, Julia, 'Visual Basic', **CoffeeScript**, Eiffel
Cluster5	Azul: demostraciones automáticas Verde: declarativo o manejo intensivo de datos	'OpenEdge ABL', **Coq**, Rouge, Thrift, **PLSQL**, Vue, Alloy, Eagle, Roff, 'GCC Machine Description', Brainfuck, 'Jupyter Notebook', 'Standard ML', **Idris**, Mako, Racket, **SQLPL**, 'Vim script', F#, 'Inno Setup', Smarty, **PLpgSQL**, **Nginx** *(Servidor de alto rendimiento), HCL, 'AGS Script', Self, Logos, XML, Cuda, 'Ragel in Ruby Host', **Puppet**,
Cluster6		SystemVerilog, PostScript, FreeMarker, Smalltalk, Slash, Hy, NewLisp, Fancy, UrWeb, Cycript,

Cluster7	Negro: web y web-framework	**CSS, HTML, Python, JavaScript, Shell**
Cluster8		'Game Maker Language', Fortran, Processing, Bison, Gnuplot, CLIPS, M, Prolog, Handlebars,
Cluster9		'Graphviz (DOT)', QML, Smali, Max, ASP, ApacheConf, KiCad, PureBasic, Myghty, SaltStack, Nu, Clarion, QMake, Clean, HLSL, Harbour, RenderScript, XC, Parrot, IDL, nesC, Genshi, NetLogo, 'Component Pascal', MAXScript, SourcePawn, AspectJ, ANTLR, Mercury, MoonScript, Befunge, Nemerle, UnrealScript, AppleScript, Xtend, FLUX, Arduino, Scilab, PureScript, Stan, Agda, Stata, 'POV-Ray SDL', Jasmin, PigLatin, XSLT, Brightscript, XS, Bro, Nim, Hack, 'Gettext Catalog', Crystal, EmberScript,
Cluster10		'Module Management System', CMake, Swift, SAS, LLVM, Objective-C++, NSIS, VimL,

Tabla N° 2: clusters generados para K=10 con EM aplicado al conjunto de datos N° 1

Con respecto a la tabla N° 2 se destaca como la gran parte del cluster N° 1 es ocupado por lenguajes funcionales como los identificados en la tabla N° 1. De igual manera, los lenguajes que compilan a JavaScript se separaron del cluster que contenía a los funcionales y quedaron en el cluster N° 4. El cluster N° 7 sólo contiene lenguajes identificados con la programación web, tanto del lado del cliente como del lado del servidor. El cluster N° 3 contiene a buena parte de los lenguajes usados para la programación de sistemas de bajo nivel.

Cluster1	Púrpura: web con características funcionales y compilan a JavaScript. Es notable como las características funcionales de estos lenguajes generan confusión con Haskell usando 20 clusters. Cian: circuitos	Lua, **Dart, LiveScript, Haxe**, Clojure, **JSONiq, Elm,** Rust, ColdFusion, Go, Vala, Cucumber, Cirru, **Verilog**, Liquid, LSL, PowerShell, **VHDL**, Lean, Haskell, Mask, ABAP, OpenSCAD, Forth, Groovy, C#, Tcl, Nix, XQuery, 'Protocol Buffer', COBOL, Pascal, Ada, AutoHotkey, 'Visual Basic', **CoffeeScript**, Eiffel
Cluster2		PostScript, Smalltalk, Hy, NewLisp
Cluster3		Racket, SAS
Cluster4		Java, GLSL, Ruby, OCaml, Io, Erlang, ActionScript
Cluster5		'Jupyter Notebook',Prolog

Cluster6		TypeScript
Cluster7	Verde: declarativo o manejo intensivo de datos	'OpenEdge ABL', Myghty, Brainfuck, XC, IDL, Genshi, **SQLPL**, AspectJ, ANTLR, **PLpgSQL**, **Nginx** *(Servidor de alto rendimiento), Arduino, Stan, HCL, Brightscript, M, Cuda, **Puppet,**
Cluster8		Perl, TeX, Objective-C, Assembly,
Cluster9		'Module Management System', CMake, LLVM, Objective-C++, VimL
Cluster10	Azul: demostraciones automáticas marrón:multimedia	GDScript, GDB, **Coq**, Rouge, **'Pure Data', 'Game Maker Language'**, Alloy, PicoLisp, Terra, 'GCC Machine Description', SystemVerilog, Harbour, **Idris**, Slash, nesC, **SuperCollider**, 'Vim script', Gnuplot, MoonScript, AppleScript, Xtend, Fancy, UrWeb, **Agda**, Stata, Self, E, XML, 'Ragel in Ruby Host', Cycript, EmberScript,
Cluster11	Magenta oscuro: funcional Rojo: cómputo científico	D, **Scala**, **R**, **Elixir, Scheme**, **Matlab**, '**Common Lisp**',
Cluster12		Thrift, PLSQL, Mako, Smarty, PigLatin, XSLT,
Cluster13		'Standard ML', F#, 'Inno Setup', Logos,
Cluster14		'Graphviz (DOT)', QML, Smali, Max, Vue, KiCad, PureBasic, SaltStack, Nu, Clarion, Eagle, Roff, Clean, HLSL, RenderScript, Parrot, NetLogo, 'Component Pascal', MAXScript, Bison, Mercury, Befunge, Nemerle, FLUX, PureScript, Jasmin, 'AGS Script', CLIPS, Bro, Nim, Hack, Handlebars, 'Gettext Catalog', Crystal,
Cluster15	Rojo claro: diseño de lenguajes	'DIGITAL Command Language', **Lex**, Awk, Groff, DTrace, GAP, M4, Perl6, 'Emacs Lisp', Rebol, **Yacc**, Mathematica,
Cluster16	Azul: programación de sistemas	**C++**, Batchfile, **C**,
Cluster17	Negro: web del lado del cliente y del servidor	**CSS, HTML, Python,** Makefile**, JavaScript, Shell**
Cluster18	Negro: web del lado del servidor	**PHP, ApacheConf**, Julia
Cluster19		Fortran, Processing, Kotlin, FreeMarker, Swift, NSIS

Cluster20		ASP, QMake, SourcePawn, UnrealScript, Scilab, 'POV-Ray SDL', XS

Tabla N° 3: clusters generados para K=20 con EM aplicado al conjunto de datos N° 1

De la tabla N° 3 se pueden apreciar los tipos de lenguajes que siguen manteniendo su cohesión. A este punto, se puede presumir que los clusters empiezan a reflejar las particularidades de los datos y no son generalizables. Sin embargo, en el cluster N° 17 se siguen encontrando juntos los lenguajes asociados con la programación web. Los lenguajes funcionales en el cluster N° 11 fueron separados de Clojure y Haskell, pues éstos se encuentran en el cluster N° 1. En este mismo cluster se encuentran los lenguajes que compilan a JavaScript. Se puede presumir que el hecho de usar uno de estos tipos de lenguajes despierta el interés en usar Haskell y Clojure, pues lenguajes como Dart, Elm y LiveScript tienen características fuertemente funcionales.

Cluster1		**Java, Scala**
Cluster2		SystemVerilog, PostScript, GAP, Smalltalk, Slash, Perl6, Hy, NewLisp, Fancy, UrWeb, Mathematica, Cycript
Cluster3		'Graphviz (DOT)', QML, Smali, Coq, Rouge, Max, KiCad, PureBasic, 'Game Maker Language', Alloy, Eagle, Clean, Harbour, Idris, RenderScript, XC, IDL, nesC, NetLogo, 'Component Pascal', MAXScript, AspectJ, ANTLR, F#, 'Inno Setup', Mercury, PLpgSQL, MoonScript, Befunge, Nemerle, AppleScript, FLUX, Arduino, PureScript, Agda, Stata, HCL, Jasmin, 'AGS Script', Self, Brightscript, Logos, Bro, Nim, XML, 'Gettext Catalog', Cuda, 'Ragel in Ruby Host', Crystal
Cluster4		Fortran, Kotlin, FreeMarker, SQLPL, Xtend
Cluster5		'Module Management System', CMake, Swift, SAS, Gnuplot, LLVM, Objective-C++, NSIS, CLIPS, VimL
Cluster6	Magenta oscuro: funcional Púrpura: web con características funcionales y compilan a JavaScript Amarillo: Adobe	PHP, Lua, **Dart, LiveScript, Haxe**, **Clojure**, **JSONiq, Elm**, Rust, **ColdFusion**, Go, Vala, Cucumber, Cirru, Verilog, Liquid, GLSL, LSL, PowerShell, VHDL, Lean, **Haskell**, Mask, ABAP, Ruby, OpenSCAD, Forth, Groovy, **OCaml,** C#, Io, Tcl, Nix, XQuery, 'Protocol Buffer', **Erlang**, **ActionScript,** COBOL, Pascal, Ada, AutoHotkey, Julia, 'Visual Basic', **CoffeeScript**, Eiffel
Cluster7		'Emacs Lisp'

Cluster8		Nu, Parrot, Prolog
Cluster9		Brainfuck, 'Standard ML', Racket, Stan
Cluster10		Roff, HLSL, M
Cluster11		GDScript, GDB, 'DIGITAL Command Language', **Lex**, 'Pure Data', Awk, PicoLisp, Groff, DTrace, Terra, M4, SuperCollider, Rebol, **Yacc,** E
Cluster12	Magenta oscuro: funcional Rojo: cómputo científico	D, TeX, **R**, **Elixir, Scheme**, **Matlab**, **'Common Lisp'**
Cluster13		Vue, SaltStack, 'Vim script', Nginx, Hack, Handlebars
Cluster14		TypeScript
Cluster15		Thrift, PLSQL, Myghty, Mako, Genshi, Smarty, PigLatin, XSLT
Cluster16	Negro: web del lado del cliente y del servidor	**CSS, HTML, Python,** Makefile, **JavaScript, Shell**
Cluster17	Azul: desarrollo de sistemas	Perl, **C++,** Batchfile, **Objective-C, Assembly, C**
Cluster18		'OpenEdge ABL', 'Jupyter Notebook', Processing, EmberScript
Cluster19		ApacheConf, Clarion, 'GCC Machine Description', Puppet
Cluster20		ASP, QMake, SourcePawn, Bison, UnrealScript, Scilab, 'POV-Ray SDL', XS

Tabla N° 4: clusters generados para K=20 con K-Means aplicado al conjunto de datos N° 1

La tabla N° 4 muestra los resultados del agrupamiento para 20 clusters con K-Means. Este algoritmo sigue agrupando a los lenguajes HTML, CSS, Python y Shell en el cluster N° 16. También agrupa a varios lenguajes con características funcionales en el cluster N° 6. A diferencia del algoritmo EM, este no diferencia los distintos usos de los lenguajes con características funcionales que compilan a JavaScript de aquéllos que normalmente son usados en entornos académicos.

3. Descripción del Segundo Experimento

De la misma manera que el experimento anterior, se consideran como atributos los lenguajes de programación. Sin embargo, las instancias representan los repositorios. Por esto, cada atributo representa la cantidad de bytes presentes en dicho repositorio.

4. Resultados del Segundo Experimento

En este caso, se probaron los algoritmos EM y K-Means con el conjunto de datos de 8127 repositorios . Sin embargo, en ambos casos casi la totalidad de las instancias se agruparon en el mismo cluster.

5. Descripción del Tercer Experimento

Para este experimento se utilizó el último conjunto de datos con 2335 usuarios y sus repositorios con dos usuarios raíz, en fin de tener la mayor cantidad de data para hacer una buena representación de la población de usuarios de github.

En esta serie de corridas se usó la probabilidad condicional (P(A|B)) de que un usuario use un lenguaje dado que usa otro para buscar similitudes y conjuntos de lenguajes que son usados juntos. Nótese que esta probabilidad no es simétrica, es decir P(C++ | C) != P(C | C++). Se elaboró una tabla con estas probabilidades para los 10 (por motivos de espacio) lenguajes más usados (por usos en repo) para tener una intuición de que lenguajes son usados con que otros.

Además, usando la correlación de Pearson, se elaboró un gráfico mostrando las correlaciones entre los 20 lenguajes más populares (por usos en repo) en la data recolectada con el fin nuevamente de poder analizar la data de mejor manera. También se utilizó el coeficiente de correlación de spearman (que intenta usar una funcion monotona en lugar de lineal para la correlación entre datos) pero ésta sólo repetía los resultados de la correlación de pearson con mayor intensidad, por lo que fue descartada.

Usando las tablas de probabilidad condicionales entre cada par de lenguajes se creó una función de distancia para permitir el clustering (Ya que hay muy pocos algoritmos que permitan clustering con distancias asimetricas). Ésta función de distancias toma para cada par X,Y el máximo valor entre 1 - P(X|Y) y 1 - P(Y|X). Se toma 1 - P(X|Y) ya que se necesitan distancias y la probabilidad P(X|Y) es mayor entre más “similares” sean X y Y. Se usa el maximo en fin de reducir el efecto que puedan tener lenguajes que sean muy utilizados como javascript.

Una vez conseguida la matriz de distancias, se usó cluster jerárquico (con link simple) con una cantidad reducida de lenguajes para realizar un dendrograma de los clusters encontrados y luego se usó DBSCAN (Eps = 0.5, mínimo agrupamiento = 2) con todos los lenguajes con el fin de que DBSCAN pudiera decidir la cantidad de clusters.

6. Resultados del Tercer Experimento

Primero se tiene la tabla de probabilidades condicionales, en esta se muestra la probabilidad de la fila dada la columna, es decir, P(PHP|CSS) = 0.93 y P(CSS|PHP) = 0.53.

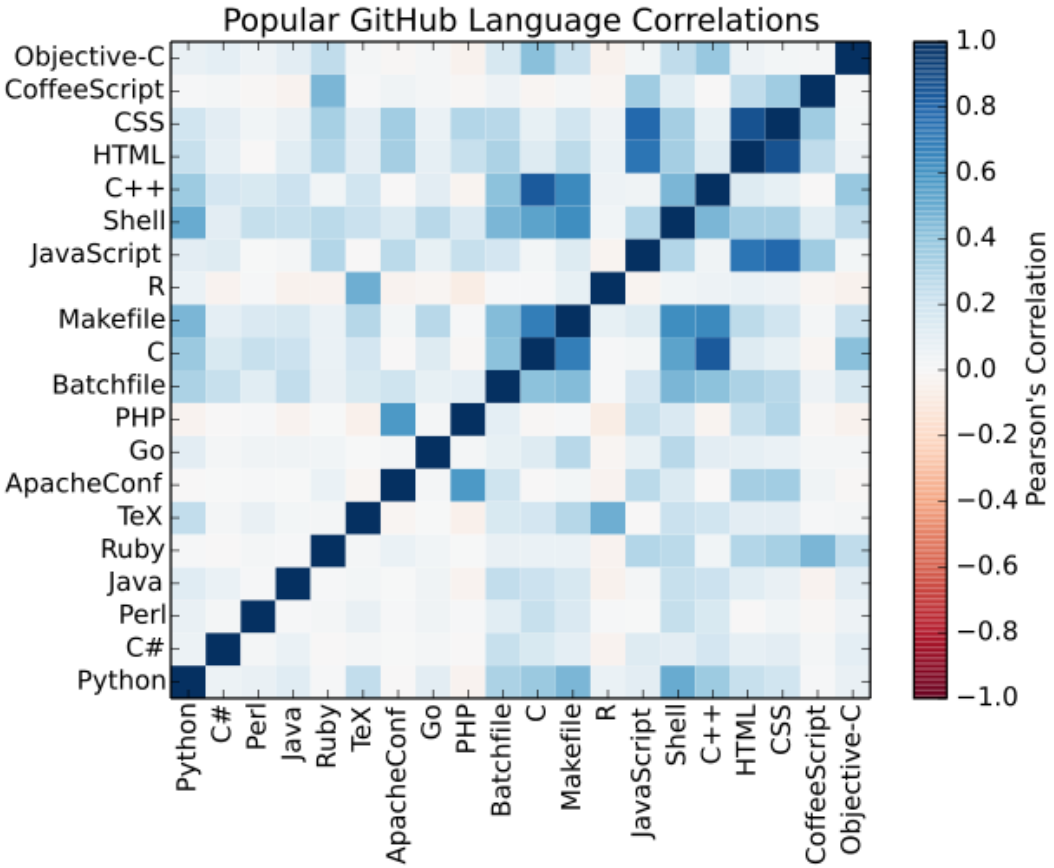
La principal observación que podemos hacer de esta tabla es que la mayoría de los usuarios llegan a usar lenguajes como HTML, CSS y JS, por lo que la probabilidad de un lenguaje dado que usen estos tres lenguajes anteriores tiende a ser alta y no nos dice mucho de la relación entre ellos.

Lenguaje	PHP	CSS	Makefile	C	Java	HTML	JS	Shell	Ruby	Python
PHP	1.00	0.93	0.55	0.47	0.54	0.89	0.96	0.82	0.59	0.68
CSS	0.53	1.00	0.54	0.48	0.52	0.91	0.96	0.80	0.58	0.69
Makefile	0.53	0.92	1.00	0.69	0.62	0.92	0.94	0.94	0.65	0.86
C	0.48	0.88	0.74	1.00	0.67	0.86	0.90	0.91	0.64	0.83
Java	0.51	0.87	0.62	0.62	1.00	0.86	0.89	0.84	0.59	0.75
HTML	0.52	0.94	0.56	0.49	0.53	1.00	0.94	0.80	0.57	0.70
JS	0.53	0.93	0.54	0.48	0.52	0.88	1.00	0.79	0.57	0.67
Shell	0.52	0.89	0.62	0.56	0.56	0.87	0.92	1.00	0.61	0.75
Ruby	0.54	0.92	0.61	0.56	0.56	0.88	0.95	0.88	1.00	0.72
Python	0.50	0.89	0.66	0.59	0.58	0.88	0.90	0.87	0.59	1.00

Matriz muestra P(fila|columna) para los lenguajes más populares

A continuación se muestra la imagen que contiene las correlaciones entre los 20 lenguajes más populares en la dataset usada.

En esta imagen lo principal que se tiene que ver es la alta correlación entre javascript y otros lenguajes web, igual que la alta correlación entre C y C++, también se puede ver que R, siendo un lenguaje utilizado principalmente por matematicos, estadisticos, entre otros, tiene muy poca relación con la mayoría de los otros lenguajes excepto TeX, que es un sistema de tipografía muy usado para crear informes, trabajos, entre otros por las distintas carreras científicas.

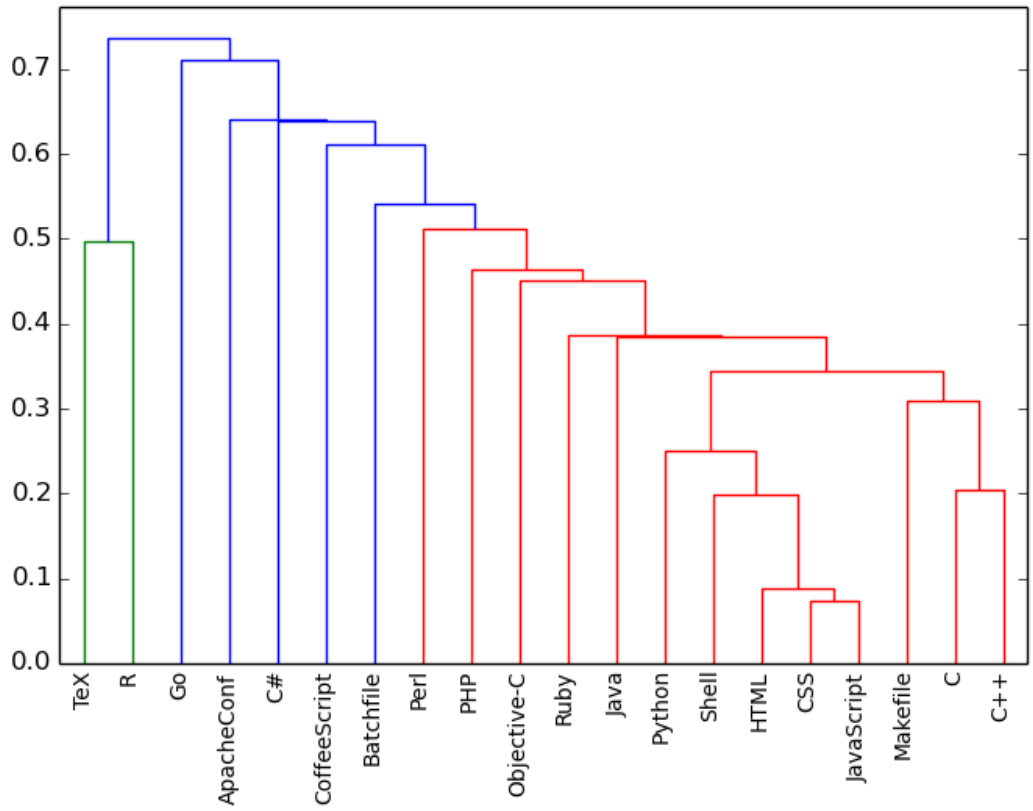


Correlaciones entre los 20 lenguajes más usados en el dataset.

Una vez visualizada la data se empezó a intentar usar clustering, primeramente se realizó el clustering con una baja cantidad de datos en fin de poder visualizar el resultado, por esto se creó un dendrograma con los 20 lenguajes más usados en el dataset para poder visualizarlo sin problemas.

En este dendrograma podemos ver fácilmente que se mantienen las correlaciones obtenidas anteriormente, con C y C++ estando juntos, javascript, HTML y CSS también y R y TeX agrupados al final del dendrograma. También es interesante ver que se agrupan todos los lenguajes como Ruby, python y java con C y C++ y también con Javascript, HTML y CSS, mostrando que aunque todos estos lenguajes tengan diferentes usos, todavia se usan juntos lo suficientemente frecuente como para que el algoritmo los coloque en el mismo cluster.

Para el algoritmo aglomerativo se utilizó la matriz definida anteriormente (Dist(X,Y) = Max(1-P(X|Y),1-P(Y|X))) y enlace simple.



Dendrograma de agrupamiento de los 20 lenguajes más populares

Luego de este dendrograma se utilizó DBSCAN para intentar agrupar en clusters a todos los lenguajes, se utilizó DBSCAN debido a que no se necesita escoger el número de clusters y que el mismo descarta lenguajes que no se puedan utilizar en clusters.

Lo principal que se puede ver en los resultados de estos clusters es la agrupación de los lenguajes más comunes que vemos en un solo cluster, además, podemos ver que TeX y R aparecen como un cluster igual que Lex y Yacc (Lexer y parser de c).

Además de los tres clusters con lenguajes semi conocidos, encontramos otros clusters con lenguajes con poco uso que nos lleva a pensar que estos lenguajes son usados muy pocas veces pero de manera conjunta en el conjunto de datos, esto hace que su probabilidad condicional sea 1 (de hecho, la distancia entre PicoLisp y GDScript es 0), y por lo tanto estén muy cercanos, para cualquier futuro estudio se debería tratar esto con un factor que disminuya la importancia de lenguajes con poco uso, o con la eliminación de los mismos completamente.

Debido a la gran cantidad de lenguajes cuyo poco uso afecta la matriz de distancias, no se pudo encontrar una manera de crear clusters más específicos o informativos sin eliminar lenguajes, por lo que se muestran estos con el motivo de mostrar el efecto que estos tienen en la agrupación.

Cluster	Observaciones	Contenido
Cluster 0	Lenguajes con muy poco uso.	Agda, Turing, Cool, J, Oz, Golo, ooc, XProc, Pike, Modula-2, Tea, Ioke, LOLCODE, Opa, Boo, Idris, Fantom, Lasso, Omgrofl, Befunge, Squirrel, Factor, PureScript, Csound, Dogescript
Cluster 1	Lenguajes con poco uso pero reconocidos, ADA y COBOL son lenguajes bastante viejos con sus usos específicos.	JSONiq, Io, VHDL, Nix, Vala, XQuery, Forth, ABAP, Verilog, COBOL, Eiffel, Mask, Ada, Lean, Cirru, OpenSCAD, AutoHotkey, Liquid, Elm, LSL, Haxe
Cluster 2	Lenguajes “Principales” usados comúnmente, DBSCAN los agrupó todos juntos	Ruby, HTML, Java, PHP, Shell, Python, Objective-C, CSS, JavaScript, C++, C, Makefile
Cluster 3	Muy poco uso.	Nemerle, BlitzBasic
Cluster 4	Muy poco uso.	IGOR Pro, ChucK, KRL
Cluster 5	Muy poco uso.	UrWeb, Cycript, Fancy
Cluster 6		SAS, Module Management System
Cluster 7		REXX, Ragel
Cluster 8	Herramientas de python, muy poco uso	Myghty, Genshi
Cluster 9	Muy poco uso, distancia entre ellos es 0	PicoLisp, GDScript
Cluster 10	Lenguajes usados por estadísticos y matemáticos comúnmente, por lo que están juntos al ser usados por la misma clase de personas.	TeX, R
Cluster 11	Parser y Lexer de C, muy pocas veces se usa uno sin el otro	Lex, Yacc

6. Conclusiones

Los resultados obtenidos indican que el agrupamiento a nivel de lenguajes y de usuarios es el más efectivo. Con el primer experimento se pudieron encontrar ciertos perfilamientos de usuarios. Algunos de ellos eran de esperar, como el cluster N° 7 de la tabla N° 2. Este cluster contiene a los lenguajes más usados en la programación web del lado del cliente. El hecho de que incluya a Python hace pensar en su frecuente uso del lado del servidor con *frameworks* como Django, Web2py y Flask. Sin embargo, también se encontraron clusters inesperados. Como ejemplo de esto se puede mencionar al cluster N° 6 de la tabla N° 4 que contiene principalmente lenguajes con características funcionales pero que no se agrupan con los otros funcionales, como en el cluster N° 1 de la tabla N° 2. Lo que diferencia en este caso a ambos tipos de lenguajes es su comunidad de uso y su objetivo principal. Los lenguajes marcados en púrpura del cluster N° 4 de la tabla N° 2 son usados principalmente como reemplazo del lenguaje JavaScript, mientras que los demás lenguajes con características funcionales son usados en entornos académicos principalmente.

Por otra parte, los resultados obtenidos en el tercer experimento muestran resultados similares, aunque de manera mucho más clara. Varios ejemplos de esto se pueden notar en el dendrograma del agrupamiento de los veinte lenguajes más populares. En él se confirma el agrupamiento de los lenguajes más usados en la programación web del lado del cliente con HTML, CSS y JavaScript así como su cercanía a Python por su uso en frameworks populares de desarrollo web del lado del servidor. Un caso interesante es la cercanía entre el lenguaje estadístico R y TeX. Ambos, aunque radicalmente distintos, suelen ser usados por la misma comunidad científica para llevar a cabo sus investigaciones y proyectos. Otro caso que es de notar y que se esperaba es la relación vista en ambos experimentos entre Makefiles, C y C++. Esto se puede apreciar fácilmente en el dendrograma.

Una observación interesante que se puede extraer de la matriz de correlaciones entre los veinte lenguajes más usados en el conjunto de datos es la relativa ausencia de correlaciones negativas. Esto refleja el potencial de uso de cualquier combinación de lenguajes. Los lenguajes con mayores cantidades de correlaciones negativas son PHP y R. Esto se puede presumir por ser lenguajes muy limitados a un dominio específico.

Finalmente, tal como se puede ver en la tabla N° 0, la cantidad de lenguajes muy usados es poca en comparación con la cantidad de lenguajes poco usados, que es grande. Esto implica que la información confiable que se pueda extraer de dichos lenguajes a partir de los agrupamientos del primer experimento es poca. En el caso del tercer experimento, al usar la probabilidad condicional

sin tomar en cuenta el uso de cada par de lenguaje, los lenguajes con muy poco uso pueden llegar a tener distancias muy bajas entre ellos si sus pocos usos son en conjunto uno con el otro, esto causa que el uso de estos lenguajes sea muy dañino para cualquier agrupación que se quiera lograr con la función de distancia usada.

6. Referencias

[1] arXiv:1603.00431 [cs.PL]
[2] Aggarwal, C., & Reddy, C. (2014). Data clustering (1st ed.). Boca Raton: CRC Press.
[3] Marsland, Stephen. Machine Learning. 2nd ed.
[4] Tan, P., Steinbach, M., & Kumar, V. (2005). Introduction to data mining (1st ed.). Dorling Kindersley: Pearson.