Εξαγωγή σχεσιακών δεδομένων από την Wikipedia

Χρήστος Περιβολαρόπουλος

Sunday 21 February 2016

class= στη αρχη τάξη μετα κατηγορία

category= κατηγορία

value= τιμή και αξία

Ο χρόνος των ρημάτων είναι πρώτος πληθυντικός ΠΧ καναμε, βρήκαμε και όχι βρισκεις

Περιεχόμενα

I Περίληψη 2

II Εισαγωγή 4

1 START 5

2 Omnibase, wikipedia, wikipediabase και wikipedia mirror 7

III Wikipediabase 8

3 Λειτουργικότητα10

4 Getting started 15

5 Αρχιτεκτονική 17

6 Provider/Acquirer μοντέλο 29

7 Τεστάρισμα 33

8 Συνώνυμα 39

9 Backend Βάση Δεδομένων 45

10 Πηγές Δεδομένων48

11 Date parser 50

1

IV WikipediaMirror 53

12 mediawiki stack επισκόπηση55

13 Εγκατάσταση 58

14 Εργαλεία 60

15 Mediawiki Extensions 63

16 Dumps 65

17 The xerces bug 66

18 Automation 76

19 Performance 82

20 Appendix (script sources) 83

V Related CSAIL projects 100

VI Conclusion 101

2

Part I

Abstract

3

MiT InfoLab’s START (SynTactic Analysis using Reversible Transforma-

tions) είναι το πρώτο σύστημα παγκοσμίως που βασίζεται σε ερώτηση και απάντηση. Για την πρόσβαση σε περισσότερες πηγές δεδομένων χρησιμοποιεί

Omnibase , την " εικονική βάση δεδομένων " που παρέχει

πρόσβαση σε πολλαπλές πηγές στο διαδίκτυο . Αναπτύξαμε Wikipedia-Base

για την παροχή παρόμοιου τρόπου δεδομένων όπως το Omnibase για το START. Με στόχο να αποκτηθεί για να αποκτηθεί πρόσβαση σε μη δομημένεςκαι ημι - δομημένες πληροφορίες στη Wikipedia . Στο πλαίσιο αυτού του στόχου επίσης δημιουργήσαμε το wikipedia - -mirror, ένα πρόγραμμα που δημιουργεί κλώνους της Wikipedia κλώνους που μπορεί να τρέχουν τοπικά , με αποτέέσμα να παρέχει έλεγχο και απεριόριστη πρόσβαση στο σύνολο των δεδομένων της wikpedia χωρίς να εξαρτάται από χρήση ή την κατάχρηση της wikipedia.org .

4

Part II

Introduction

5

Chapter 1

START

The START Natural Language System είναι ένα σύστημα λογισμικού που έχει σχεδιαστεί για να

απαντά σε ερωτήσεις που τίθενται σε αυτό σε φυσική γλώσσα.. Το START σαρώνει τις

εισερχόμενες ερωτήσεις, τις ταυτίζει με τα ερωτήματα που δημιουργούνται από τα parse trees, τις αναλύει σύμφωνα με την γνωστική του βάση και τελικά παρουσιάζει τα κατάλληλα τμήματα πληροφοριών για το

χρήστη. Με τον τρόπο αυτό , το START παρέχει στους ανεκπαίδευτους χρήστες γρήγορη πρόσβαση σε

γνώση που σε πολλές περιπτώσεις μπορεί να είναι χρονοβόρα ακόμα και για τον ειδικό να την αποκτήσει

START (SynTactic Analysis using Reversible Transformations) δημιουργήθηκε από τον Dr Boris Katz στο Artificial Intelligence Laboratory του MIΤ. Επί του παρόντος,

το σύστημα υφίσταται περαιτέρω ανάπτυξη από το InfoLab Group, led by

με επικεφαλής τον Dr Boris Katz . Το START για πρώτη φορά συνδέθηκε με το World Wide Web το Δεκέμβριο,1993 , και με τις διάφορες μορφές του έχει μέχρι σήμερα απαντήσει σε εκατομμύρια ερωτήσεις

από χρήστες σε όλο τον κόσμο .

Μια βασική τεχνική που ονομάζεται"natural language annotation" " φυσική γλώσσα σχολιασμού " βοηθά το START να συνδέσει την αναζητούμενη πληροφορία με τις πηγές πληροφόρησης . Αυτή η τεχνική χρησιμοποιεί προτάσεις και φράσεις της φυσικής γλώσσας ως περιγραφές περιεχομένων που σχετίζονται με τα τμήματα πληροφοριών σε διάφορες granularities .

Ένα τμήμα των πληροφοριών ανακτάται όταν ο σχολιασμός του ταιριάζει με την εισαγόμενη

ερώτηση. Αυτή η μέθοδος επιτρέπει στο START να χειριστεί μεγάλη ποικιλία μέσων , συμπεριλαμβανομένων

των κειμένων, διαγραμμάτων, εικόνων , βίντεο και ήχων, συνόλων δεδομένων, ιστοσελίδων και

άλλων.

Η επεξεργασία της φυσικής γλώσσας του START αποτελείται από δύο

ενότητες που μοιράζονται την ίδια γραμματική . Η μονάδα κατανόησης

αναλύει το αγγλικό κείμενο και παράγει μια γνωστική βάση που κωδικοποιεί πληροφορίες που

βρίσκονται στο κείμενο . Λαμβάνοντας υπόψη ένα κατάλληλο τμήμα της βάσης δεδομένων, η

μονάδα παραγωγής παράγει αγγλικές προτάσεις . Αυτές οι ενότητες σε συνδυασμό με την τεχνική της φυσικής γλώσσας σχολιασμού, δίνουν την δυνατότητα της παραγωγής φυσικής γλώσσας σε επίπεδο προτάσεων με χρήση σε υπηρεσίες πρόσβασης πληροφοριών διαφόρων πολυμέσων.

Chapter 2

Omnibase, wikipedia,

wikipediabase and wikipedia

mirror

Omnibase είναι μια " εικονική " βάση δεδομένων που παρέχει μια ενιαία διεπαφή για πολλαπλές

πηγές γνώσης στο Web , ικανή να εκτελεί τα δομημένα ερωτήματα που

παράγονται από το START. Το Omnibase αναπτύχθηκε για πρώτη φορά το 2002 , περίπου το ίδιο χρονικό διάστημα που η wikipedia έκανε την πρώτη της εμφάνιση ( 2001 ) .

Η διαδικτυακή εγκυκλοπαίδεια Wikipedia είναι μια τεράστια, συνεχώς εξελισσόμενο δίκτυο

από πλούσιες αλληλένδετες πληροφορίες σε μορφή κειμένου . Στην αυξανόμενη κοινότητα των νέων

ερευνητών και προγραμματιστών είναι μια συνεχώς εξελισσόμενη πηγή χειροκίνητα οριζόμενων

εννοιών και σημασιολογικών σχέσεω. Αποτελεί έναν απαράμιλλο και σε μεγάλο βαθμό

ανεκμετάλλευτο πόρο για την επεξεργασία φυσικής γλώσσας προγραμματισμού, διαχείριση της γνώσης,

εξόρυξη δεδομένων, και διάφορους άλλους τομείς έρευνας .

Είναι το προϊόν της συνεργασίαςεργασία εκατομμυρίων ανθρώπων . Η Wikipedia βασίζεται στο σύστημα wiki [ refrence ] ,

μια κατηγορία ιστοσελίδων που επιτρέπουν την συνεργατική τροποποίηση του περιεχομένου .

Λόγω της πολυπλοκότητας και την ιδιαίτερα αδόμητη φύση της wikipedia

αντί του omnibase backend START χρησιμοποιήσαμε μια ξεχωριστή υπηρεσία , το Wikipedi- aBase, που αποτελεί και το αντικείμενο της παρούσας διατριβής . Επίσης, για να αποφευχθεί ο κορεσμός του wikiedpedia.org χρησιμοποιήσαμε το wikipedia - mirror δημιουργώντας έναν κλώνο της wikipedia για την Wikipedi- aBase

χρησιμοποιώ.

8

Part III

Wikipediabase

9

Η WikipediaBase είναι ένα backend για το START υπεύθυνη για την παροχή πρόσωασης σε πληροφορίες που σχετίζονται με την wikipedia. Μιμείται την διεπαφή API πουπροέρχεται από το Omnibase . Η Wikipediabase ξαναγράφει δυο φορές. Η αρχική έκδοση ήταν γραμμένη σε Java . Στη συνέχεια ξαναγραφτηκε σε Ruby διατηρώντας την αρχική αρχιτεκτονική και το σχεδιασμό και τώρα ξαναγράφεται σε python με νέο σχεδιασμό και αρχιτεκτονική.

Υπάρχουν δύο βασικοί λόγοι για αυτό:Η Python διδάσκεται ως προπτυχιακό και μεταπτυχιακό μάθημα στο MIT , και ως εκ τούτου, μια βάση κώδικα σε Python

θα κάνει την έναρξη των νέων φοιτητών του ΜΙΤ ομαλότερη. Το πιο σημαντικό όμως είναι ότι

ενώ ο αρχικός σχεδιασμός του προηγούμενου WikipediaBase θα έπρεπε να ήταν επαρκής μεγάλωσε σε τέτοιο σημείο όπου ο κώδικας ήταν ad- hoc και δύσκολα να κατανοηθεί,

λόγος για την επέκταση

Η εφαρμογή python αρχικά γράφτηκε από τον Χρήστο Περιβολαρόπουλο

σε στενή συνεργασία με την Dr Sue Felshin και τελικά παραδόθηκε στους

Sue Felshin , Alvaro Morales και ton Michael Silver .Αργότερα και άλλοι φοιτητές έχουν

εντάχθηκαν στο έργο.

10

Chapter 3

Functionality

Στη WikipediaBase , το καθέ ( υποστηρίζεται) Wikipedia infobox ορίζεται ως τάξη ,

και κάθε ( υποστηρίζεται) μεταβλητή στο infobox ορίζεται ως ένα χαρακτηριστικό της

κάθε τάξης. Όλα τα αντικείμενα της WikipediaBase ανήκουν κληρονομικά στην υπερκλάση

wikibase- term, η οποίο υποστηρίζει τα χαρακτηριστικά IMAGE-DATA, SHORT-ARTICLE,

URL, COORDINATES, PROPER, και NUMBER.

Οι εντολές της WikipediaBase και οι τιμές επιστροφής τους χρησιμοποιούν Lisp -like κωδικοποίηση.

Η WikipediaBase παρέχει τις ακόλουθες λειτουργίες:

1. get

Δεδομένης μιας class, όνομα αντικειμένου, and typed attribute, η τιμή επιστροφής είναι μια lisp-readable μορφής. Compare Omnibase’s get operation.

Valid attribute typecodes είναι:code (για ένα attribute όνομα όπως στοinfobox

wiki markup) και :rendered ( για ένα attribute όνομα στο rendered

form από το infobox).

(a) Types

Τα Scripts mπρέπει να επιστρέφουν μια λίστα απόtyped values. Valid value typecodes

είναι:

i. :HTML

Ένα string προσαρμοσμένο για rendering σαν paragraph level HTML. Το

string πρέπει να είναι escaped για lisp, εννοώντας quoted, και με

double quotes και backslashes escapedμε backslashes.

Το string δεν απαιτείται να περιέχει contain κανέναν HTML κώδικες.

Για παράδειγμα:

11

(get " wikipedia -sea " " Black Sea" (: code "

AREA "))

=> ((: html "436 ,402 km2 (168 ,500 sq mi)"))

(get " wikipedia - president " " Bill Clinton "

(: code " SUCCESSOR "))

=> ((: html " George W. Bush "))

(get " wikipedia - president " " Bill Clinton "

(: rendered " Succeeded by "))

=> ((: html " George W. Bush "))

ii. :YYYYMMDD

Οι Parsed ημερομηνίες αντιπροσωπεύονται σαν αριθμοί, χρησιμοποιώντας τον τύποYYYYM-MDD με αρνητικούς αριθμούς αντιπροσωπεύονται οι B.C. ημερομηνίες.

(Οι Unparsable ημερομηνίες αντιπροσωπεύονταισαν HTML strings χρησιμοποιώντας το

:HTML typecode.)

(get " wikibase - person " " Barack Obama " (: ID

"BIRTH - DATE "))

=> ((: yyyymmdd 19610804) )

(get " wikibase - person " " Julius Caesar " (:

ID "BIRTH - DATE "))

=> ((: YYYYMMDD -1000713) )

iii. :CALCULATED

Το Typecode για attributes υπολογισμένοι από την WikiBase βασισμένοι σε χαρακτηριστικά του άρθρου, e.g., *GENDER* and *NUMBER*.

Βλέπε παρακάτω στο Special Attributes για την ολοκληρωμένη λίστα των υπολογισμένων attributes.

iv. :CODE Deprecated old synonym for :HTML.

v. :STRING Deprecated old synonym for :HTML.

(b) Special Attributes

Μερικά χαρακτηριστικά είναι ειδικά επειδή υπολογίζονται από Wikipedi- aBase

αντί να είναι τραβηγμένα από infoboxes ή δεν παρέχον ται άμεσα.

που απέφερε άμεσα . Αυτά τα χαρακτηριστικά θα πρέπει να είναι ειδικά για

wikibase- term , wikibase - person , και wikipedia - paragraphs.

i. SHORT-ARTICLE, wikibase-term

Η πρώτη παράγραφος του άρθρου ή αν η πρώτη παράγραφος είναι μικρότερη από 350 χαρακτήρες τότε επιστρέφει την πρώτη παράγραφο έτσι ώστε το άθροισμα των χαρακτήρων της πρώτης παραγράφου να είναι τουλάχιστον 350 χαρακτήρες.

ii. URL, wikibase-term

Επιστρέφει το URL του άρθρου ως ((:url URL))

iii. IMAGE-DATA, wikibase-term

Επιστρέφει μια λίστα από URLs εικόνων στο περιεχόμενο του άρθρου in the article content (αποκλείει εικόνες που είναι στη σελίδα αλλά εκτός του περιεχ ομένου του άρθρου). εάν δεν υπάρχουν εικόνες θα πρέπει να επιστρέφει μια κενή λίστα.

Η " καλύτερη" εικόνα πρέπει να είναι η πρώτη URL της λίστας;εάν υπάρχει εικόνα στην κορυφή του

infobox, αυτή θεωρείται η καλύτερη εικόνα, διαφορετικά είναι η πρώτη εικόνα που εμφανίζεται οπουδήποτε στο άρθρο. Εαν δεν υπάρχει caption, η τιμή του caption παραλείπεται

π.χ., προτιμότερο((0 "Harimau\_Harimau\_cover.jpg"))

από ((0 "Harimau\_Harimau\_cover.jpg" "")).

iv. COORDINATES, wikibase-term

Χαρακτηριστικά που δίνονται στο άρθρο υπολογιζόμενα από το γεωγραφικό πλάτος και το μήκος ή, εφόσον κανένα δεν μπορεί να βρεθεί, το infobox . Η τιμή είναι ένας

κατάλογος του πλάτους και μήκους , π.χ. , ( ( : συντεταγμένες γεωγραφικού πλάτους

γεωγραφικό μήκος))

v. BIRTH-DATE, wikibase-person

Λαμβάνονται από το infobox ή αν δεν βρεθεί, λαμβάνονται από το άρθρο,

ή αν δεν βρεθεί, από τις πληροφορίες της κατηγορίας του άρθρου.

Βασίζεται πάντα στην πρώτη ημερομηνία γέννησης που εντοπίσθηκε και ταιριάζει σε μια από τις διάφορες υποστηριζόμενες μορφές. Αν αυτό το χαρακτηριστικό έχει μια τιμή,

τότε το αντικείμενο θεωρείται ότι είναι ένα πρόσωπο με αξία

στην ιδιότητα ΦΥΛΟ ( βλέπε παρακάτω ) .

Η τιμή μπορεί να είναι μια a parsed or unparsed date. Parsed dates αντιπροσωπεύονται ως αριθμούς , χρησιμοποιώντας τη μορφή YYYYMMDD χρησιμοποιώντας αρνητικούς αριθμούς για τις ημερομηνίες Π.Χ.

Unparsed dates are strings.

vi. DEATH-DATE, wikibase-person

Λαμβάνονται με παρόμοιο τρόπο όμως το BIRTH-DATE. Επιστρέφει την ίδια τιμή όπως

BIRTH-DATE,εκτός αν το πρόσωπο ζει, τότε βγάζει λάθος τιμή με διευκρίνηση

"Currently alive".

vii. GENDER, wikibase-person

Υπολογίζεται από το περιεχόμενο της σελίδας βασιζόμενο στα heuristics όπως ο αριθμός των ανδρικών ή των θηλυκών αντωνυμιών που χρησιμοποιούνται στο κείμενο

viii. NUMBER, wikibase-term

Υπολογίζεται από το περιεχόμενο του κειμένου με βάση τα χαρακτηριστικά όπως ο αριθμός των φορών που ο τίτλος της σελίδας εμφανίζεται στον πληθυντικό. Έχει αξία για όλα τα αντικείμενα.

Returns #t if many, #f if one.

A. PROPER, wikibase-term

Υπολογίζεται από το περιεχόμενο του κειμένου με βάση τα χαρακτηριστικά όπως ο αριθμός των φορών που ο τίτλος της σελίδας εμφανίζεται με κεφαλαία γράμματα όταν δεν είναι στην αρχή της σελίδας.. Έχει αξία για όλα τα αντικείμενα.

Returns #t if proper and #f if not.

2. get-classes

Δεδομένου του ονόματος ενός αντικειμένου , επιστρέφει ένας κατάλογος με όλες τις classes που ανήκει το αντικείμενο, με τις classes να αντιπροσωπεύονται ως lisp-readable strings. Παραδοσιακά τα ονόματα των τάξεων δίνονται με μικρά γράμματα χωρίς να είναι απολύτως απαραίτητο.

E.g.,

(get - classes " Cardinal ( bird )")

=> (" wikibase - term " " wikipedia - paragraphs " "

wikipedia - taxobox ")

(get - classes " Hillary Rodham Clinton ")

=> (" wikibase - term " " wikipedia - paragraphs " "

wikibase - person " " wikipedia - officeholder " "

wikipedia - person ")

3. get-attributes

Δεδομένου του ονόματος μιας τάξης , επιστρέφει ένας κατάλογος με όλα τα γνωρίσματα της τάξης

( δηλαδή , όλες οι μεταβλητές που τα υλοποιεί infobox ) , όπως lisp-

readable strings. Επίσης , μερικές φορές δεδομένο είναι το χαρακτηριστικό να είναι αναγνώσιμο από άνθρωπο και/ή η τιμή του τύπου του κώδικα για το χαρακτηριστικό.

Τα ονόματα των χαρακτηριστικών δίνονται σε κεφαλαία γράμματα, αλλά αυτό δεν είναι

αποτελεί απόλυτη απαίτηση .

E.g.,

(get - attributes " wikipedia - officeholder " " Barack

Obama ")

=> ((: CODE " TERM\_END3 " : VALUE : YYYYMMDD ) ...)

4. Sort-symbols

Βάζοντας σε σειρά σύμβολα παίρνει κάθε σύνολο συμβόλων και τα βάζει σε σειρά δημιουργώντας υποσύνολα κατα μήκος του σχετικού άρθρου.

14

=> ((" Barack Obama ") (" Obama ( surname )"))

5. sort-symbols-named

Βάζοντας σε σειρά symbols-named παίρνει ένα συνώνυμο και ένα σύνολο συμβόλων και τα βάζει σε σειρά δημιουργώντας υποσύνολα . Εαν το symbol name είναι το ίδιο με το συνώνυμο, το ίδιο και το υποσύνολό του μπαίνουν στην αρχή.

(Αυτό πρέπει να είναι ανεξάρτητη της γραμματοσειράς, αλλά είναι?

Και τι γίνεται με τα υποσύνολα?

E.g.

(sort - symbols - named " cake " " Cake (TV series )" "

Cake ( firework )" " Cake ( film )" " Cake ( drug )"

" Cake " " Cake ( band )" " Cake ( advertisement )" "The

Cake ")

=> ((" Cake ") (" Cake ( band )") (" Cake (

advertisement )") (" Cake (TV series )")

(" The Cake ") (" Cake ( film )") (" Cake ( firework )")

(" Cake ( drug )"))

15

Chapter 4

Getting started

Η συνολική WikipediaBase βρίσκεται σε ένα git repository στο infolab’s github

orginization page

git clone git@github .com: infolab - csail / WikipediaBase

WikipediaBase εξαρτάται από πολλαπλά άλλα πακέτα python .

Ευτυχώς, η python είναι shipped όχι μονο με ένα σπουδαίο πακέτο οδηγειών αλλά επίσης με ένα μηχανισμό που ονομάζεται virtualenv το οποίο απομονώνει την εγκατάσταση των project’s dependencies από το υπόλοιπο σύστημα, έτσι αποφεύγονται προβλήματα όπως version

ή namespace collisions. Ο τρόπος που αυτό δουλεύει αποτελεσματικά είναι με το global

python installation να είναι το μισό copied και το μισό symlinked σε ένα τοπικό directory και τα dependencies να είναι εγξκαταστημένα μόνο σε ένα τοπικό sandbox.

Για να δημιουργηθεί και να ενεργοποιηθεί ένα python virtualenv:

$ virtualenv --no -site - packages py

$ . py/bin/ activate

$ which python

/the/ local / directory /py/ bin/ python

Τώρα που ασφαλώς τα έχουμε εγκαταστήσει όλα θέλουμε χωρίς να σπάσουμε κάποιο

global installation pip install -r requirements .txt

Θα χρειασθούμε μερικά επιπλέον εργαλεία για να δουλέψει η WikipediaBase:

• Postresql

16

• Redis

Η εγκατάσταση αυτών των πακέτων διαφέρει ανάλογα με ην πλατφόρμα. Και οι δύο είναι βάσεις δεδομένων. Ο σκοπός τους είναι η προσωρινή αποθήκευση επαναλαμβανόμενη υπολογισμών και

για την αποθήκευση ahead-of-time υπολογισμού , όπως το όνομα infobox σήμανσης για

καθίσταται χάρτες όνομα και συνώνυμα .

17

Chapter 5

Architecture

1. Infobox

Τα Ιnfoboxes είναι πίνακες που χρησιμοποιούνται συνήθως στη wikipedia για να παρέχουν

μια επισκόπηση των πληροφοριών σε ένα άρθρο με ένα ημι δομημένο τρόπο .

Infoboxes είναι η κύρια πηγή πληροφοριών για τη WikipediaBase

18

Figure 5.1: Ένα παράδειγμα ενός infobox

Σε mediaWiki όρους σήμανσης ένα infobox είναι ένα πρότυπο σήμανσης με

τύπο που αποδίδεται σε html , έτσι ώστε οι παρεχόμενες πληροφορίες να έχουν νόημα στο πλαίσιο που παρέχεται. Για παράδειγμα:

Στη mediawiki markup ο όρος ενός infobox είναι ένα markup template με ένα type το οποίο

γίνεταιrendered σε html έτσι ώστε οι παρεχόμενες πληροφορίες βγάζουν νόημα στο περιεχόμενο που περιέχονται. Για παράδειγμα:

{{ Infobox scientist

| name = Gerhard Gentzen

| image = Gerhard Gentzen .jpg

| image\_size =

| alt =

| caption = Gerhard Gentzen in Prague ,

1945.

| birth\_date = {{ Birth date |1909|11|24}}

19

| birth\_place = [[ Greifswald ]], [[ Germany ]]

| death\_date = {{ Death date and age

|1945|8|4|1909|11|24}}

| death\_place = [[ Prague ]], [[

Czechoslovakia ]]

| nationality = [[ Germany | German ]]

| fields = [[ Mathematics ]]

| workplaces =

| alma\_mater = [[ University of Gottingen ]]

| doctoral\_advisor = [[ Paul Bernays ]]

| doctoral\_students =

| known\_for =

| awards =

}}

will yield:

20

Figure 5.2: Ένα παράδειγμα ενός infobox

Οι τύποι του Infobox είναι οργανωμένοι με μια αρκετά εκτεταμένη ιεραρχία.

Για παράδειγμαTemplate:Infobox Austrian district είναι μια ειδική περίπτωση ενός Tem-

plate:Infobox settlement και το καθένα είναι rendered διαφορετικά. Για το συγκεκριμένο σκοπό, και για να κάνουμε mirror το markup ορίζουμε τα infoboxes, έναinfobox *I*

με attributes *ai* and values *vi* i είναι έν set of pairs (*ai; vi*) μαζί με ένα

infobox type *t*. Κάθε attribute *ai* και value *vi έχουν 2*forms:

• ένα rendered form, *ari*

και *vr*

*i* αντίστοιχα, το οποίο κάνειrendered

HTML representation και

• a markup form, *ami*

and *vm*

*i το οποίο είναι το* mediawiki markup code

αυτό αντιστοιχεί σ αυτά.

21 Ένα άρθρο μπορεί να έχει περισσότερες από ένα infoboxes, για παράδειγμα, το άρθρο για τον Bill Clinton

έχει δύο Infobox ένα για Officeholder και ένα για Infobox President.

Η κατηγορία Infobox είναι ο βασικός τύπος δεδομένων για την πρόσβαση σε πληροφορίες από το infobox ενός άρθρου . Το Infobox ,όπως και το Άρθρο , είναι αυτό που

θα χρησιμοποιήσει κάποιος όταν χρησιμοποιεί τη wikipediabase ως βιβλιοθήκη Python.

Οι μέθοδοι που παρέχονται από ένα Infobox είναι :

τύποι επειδή έχουμε ανακτήσει Infobox βασισμένοι σε ένα όνομα συμβόλου ( π.χ.όνομα της σελίδας ), ένα μοναδικό Infobox μπορεί στην πραγματικότητα να είναι μια διεπαφή για

πολλαπλά infoboxes. Υπάρχει μια ξεχωριστή μέθοδος, που βασίζεται σε αυτό, για την ανάκτηση τύπων σε μορφή κατάλληλη για το START.

Η πρόσβαση είναι δυνατή υπό τον όρο είτε ari

ή ami.

Τα κλειδιά που παρέχονται με τη χρήση του MetaInfobox ( βλέπε παρακάτω ) .

Το Infobox εξάγει σε τύπους python , δηλαδή:

• dict for *ari*

*! vr*

*i* or *ami*

*! vm*

*i*

•Το συνολικό infobox rendered, ή σε ένα markup form.

Τα Infoboxes οργανώνονται σε μια ευρεία ιεραρχία το οποίο στη WikiepdiaBase

codebase αναφέρεται ως infobox tree. Τo infobox tree ανακτάται

από τη λίστα της σελίδας wikipedia infoboxes και χρησιμοποιείται για να συνταχθεί η

οντολογία των όρων wikipedia .

2. MetaInfobox

Το MetaInfobox είναι μια υποκατηγορία του Infobox που παρέχει πληροφορίες σχετικά με το infobox , το πιο σημαντικό είναι ένας χάρτης μεταξύ γνωρισμάτων σήμανσης. Ας υποθέσουμε ότι έχουμε ένα infobox τύπου *I*, το οποία έχει τα χαρακτηριστικά έ *a*1*; :::; an* . Κάθε περίπτωση που καθορίζει αυτό το infobox *I.*

Είναι ένα infobox με όλες τις έγκυρες ιδιότητες και κάθε τιμή είναι τα

ονόματα όλων των χαρακτηριστικών που είναι ισοδύναμα με αυτά. Π.χ. Ένα infobox του τύπου

Foo που έχει έγκυρα χαρακτηριστικά A, B, C και D και A, B και C είναι

ισοδύναμο έχει ένα MetaInfobox που μοιάζει κάτι σαν:

Χαρακτηριστική Τιμή:

A !!!A!!! !!!B!!! !!!C!!!

B !!!A!!! !!!B!!! !!!C!!!

C !!!A!!! !!!B!!! !!!C!!!

D !!!D!!!

22

3. Article  
Η δομή του Άρθρου είναι υπεύθυνη για την πρόσβαση σε κάθε πόρο σχετικό με το άρθρο γενικότερα. Αυτό περιλαμβάνει τις παραγράφους , επικεφαλίδες , σήμανσηςπροέλευσης και τις κατηγορίες MediaWiki.

4. Fetcher

Το fetcher είναι μια αφαίρεση από την επικοινωνία της Wikipedia-

Base με τον έξω κόσμο. Είναι ένα μονήρες αντικείμενο που υλοποιεί

μια συγκεκριμένη διεπαφή .

Τα Fetchers οργανώνονται σε μια κληρονομική BaseFetcher. Η βασική κατηγορία για fetchers , θα επιστρέψει το σύμβολο αντί να προσπαθήσει να το επιλύσει με οποιονδήποτε τρόπο. Το Fetcher περιέχει την βασική λειτουργικότητα ενός fetcher . Θα fetch άρθρα από wikipedia.org . Είναι δυνατόν να κατευθύνει αυτό προς ένα mirror αλλά η εκτέλεση σε wikipedia - mirror είναι απαγορευτική.

CachingFetcher κληρονομεί fetcher και διατηρεί τη λειτουργικότητα , μόνο αυτό

χρησιμοποιεί Redis για την προσωρινή αποθήκευση των fetched συμβόλων . Είναι η προεπιλεγμένη fetcher

για την wikipediabase .

StaticFetcher είναι μια κλάση που υλοποιεί το interface BaseFetcher

αλλά αντί να φτάσει σε κάποια πηγή δεδομένων για τα δεδομένα η

τιμές επιστροφής είναι στατικά ορισμένες . Χρησιμοποιείται κυρίως από το

MetaInfobox για να χρησιμοποιεί τη λειτουργία του Infobox να μεταφέρει αυθαίρετες

πληροφορίες.

Από προεπιλογή , η σήμανση είναι fetched από το backend . Αν forcelive έχει οριστεί σε True τότε η σήμανση θα fetched από live wikipedia.org. Όταν οι δοκιμές τρέχουν για TravisCI , θέλουμε πάντα να χρησιμοποιούνται ζωντανά δεδομένα . Εμείς ελέγξτε αν το Travis εκτελεί δοκιμές κοιτάζοντας τη μεταβλητή WIKIPEDIABASEFORCELIVEenv.

5. Renderer

Renderers είναι μονήρεις classes που είναι χρήσιμα για την απόδοση MediaWiki markup σε HTML . Αρχικά το wikiepedia sandbox χρησιμοποιήθηκε από τη wikipediabase για την απόδοση σελίδων ,επειδή είναι ελαφρώς ταχύτερο από την API, αλλά η wikipedia - mirror ήταν πολύ αργή σε αυτό και το wikipedia.org το θεωρούσε κατάχρηση της υπηρεσίας και μπλόκαρε το IP μας.

Γι'αυτό το λόγο τελικά μεταπηδήσαμε στο API με Redis caching , το οποία

λειτουργούν αρκετά καλά , επειδή τα Renderer αντικείμενα καταλήγουν να χρησιμοποιούν μόνο το

MetaInfobox μου, το οποίο έχει ένα αρκετά περιορισμένο πεδίο εφαρμογής , και έτσι το cache να

χάνει σπάνια .

Μια ενδιαφέρουσα αδημοσίευτη πληροφορία για την κατηγορία Renderer ήταν ότι αυτός ήταν ο λόγος που ένα ζευγάρι CSAIL IPs να αποκλειστεί προσωρινά από την επεξεργασία της

wikipedia. Ενώ η wikipedia.org έχει μια πολύ επιεική πολιτική όταν

πρόκειται για την αποκλεισμό των ανθρώπων που έχουν κανει spamming τους servers , Επαναλαμβανόμενες δοκιμές της κατηγορίας Renderer με στόχευση wikipedia sandbox προκάλεσε το ip του δοκιμαστικού μηχανήμτος να απακλεισθεί προσωρινά με το σκεπτικό ότι

" η δραστηριότητα του δεν προάγει την βελτίωση της wikipedia " .

Εμείς επανατοποθετήσαμε το Renderer να χρησιμοποιηεί το wikipedia API και ποτέ δεν είχαμε ξανά πρόβλημα με την ρύθμιση της wikipedia

6. Pipeline

Κατά την επίλυση ενός ερωτήματος η WikipediaBase ενεργοποιεί έναν pipeline ενοτήτων

για να διαπιστωθεί ποιος είναι ο καλύτερος τρόπος απάντησης.

(a) Frontend

Η WikipediaBase μπορεί να χρησιμοποιηθεί ως βιβλιοθήκη αλλά ο πρωταρχικός της λειτουργία είναι ως backend στο START. Η επικοινωνία μεταξύ START και της

WikipediaBase is carried out over a plaintext telnet connection

on port {port} using EDN-like sexpressions. Το frontend χειρίζεται το δίκτυο σύνδεσης με το START, μεταφράζει τις προσλαμβανόμενες ερωτήσεις σε κλήσεις της Knowledgebase και στη συνέχεια μεταφράζει την αντίδραση της Knowledgebase σε κατάλληλα διαμορφωμένες εκφράσεις και τις επιστρέφει πίσω στο telnet connection.

(b) Knowledgebase

Η knowledgebase είναι το σημείο εισαγωγής στο rest της wikipediabase.

Χρησιμοποιεί μοτίβο Provider/Acquirer να παρέχει transaprently η frontend με αυθαίρετες μεθόδους. Οι μέθοδοι αυτοί είναι υπεύθυνοι για την επιλογή αν θέλουμε να καταλήξουμε σε classifiers ή resolvers ( ή οποιοδήποτε άλλο μηχανισμό ) για να δοθεί απάντηση στο ερώτημα . Διαθέσιμοι classifiers και resolvers γίνονται προσβάσιμοι αυτόματα στη knowledgebaseχρησιμοποιώντας τη βασική τους κατηγορία τους .

(c) Classifiers

Κάθε Classifier είναι μονήρης και υλοποιεί μια heuristic για deducing

μια σειρά από κατηγορίες ενός αντικειμένου . Ένα αντικείμενο μπορεί να αναστέλλει μηδέν ή περισσότερες κατηγορίες . Υπάρχουν ένας-δύο Classifiers διαθέσιμα επί του παρόντος

Συνήθως, ένας Classifier θα συμπεράνει μόνο αν ένα αντικείμενο

πράγματι αναστέλλει μια συγκεκριμένη κατηγορία ή όχι, αλλά αυτό δεν είναι απαραίτητο.

i. Term

Ο TermClassifier απλά αναθέτει την κατηγορία wikipedia-term .

Η Wikipediabase διαπραγματεύεται μόνο μόνο με πληροφορίες σχετικές με τη wikipedia

ii. Infobox

Το InfoboxClassifier αναθέτει σε ένα όρο την κατηγορία infobox. Για παράδειγμα η σελίδα Bill Clinton περιέχει το infobox:

{{Infobox president

|name = Bill Clinton

|image = 44 Bill Clinton 3x4.jpg{{!}}border

|office = [[List of Presidents of the United States|42nd]] [[President |vicepresident = [[Al Gore]]

|term\_start = January 20, 1993

|term\_end = January 20, 2001

|predecessor = [[George H. W. Bush]]

|successor = [[George W. Bush]]

|order1 = 40th and 42nd [[List of Governors of Arkansas|Governor of |lieutenant1 = [[Winston Bryant]]<br>[[Jim Guy Tucker]]

|term\_start1 = January 11, 1983

|term\_end1 = December 12, 1992

|predecessor1 = [[Frank D. White]]

|successor1 = [[Jim Guy Tucker]]

|lieutenant2 = [[Joe Purcell]]

|term\_start2 = January 9, 1979

|term\_end2 = January 19, 1981

|predecessor2 = [[Joe Purcell]] {{small|(Acting)}}

|successor2 = [[Frank D. White]]

|office3 = 50th [[Arkansas Attorney General|Attorney General of Arkansas]]

|governor3 = [[David Pryor]]<br>[[Joe Purcell]] {{small|(Acting)}}

|term\_start3 = January 3, 1977

|term\_end3 = January 9, 1979

|predecessor3 = [[Jim Guy Tucker]]

|successor3 = Steve Clark

|birth\_name = William Jefferson Blythe III

|birth\_date = {{birth date and age |1946|8|19}}

|birth\_place = [[Hope, Arkansas|Hope]], [[Arkansas]], [[United States|U.S.]]

|death\_date =

25

|death\_place =

|party = [[Democratic Party (United States)|Democratic]]

|spouse = {{marriage|[[Hillary Clinton|Hillary Rodham]]|October 11, |relations = ’’See [[Clinton family]]’’

|children = [[Chelsea Clinton|Chelsea]]

|parents = [[William Jefferson Blythe, Jr.]]<br>[[Virginia Clinton Kelley]]

|alma\_mater = [[Edmund A. Walsh School of Foreign Service|Georgetown University]]<br>[[University |religion = [[Baptists|Baptist]] {{small|(formerly [[Southern Baptist |signature = Signature of Bill Clinton.svg

|signature\_alt = Cursive signature of Bill Clinton in ink

|website = {{url|clintonlibrary.gov|Library website}}

}}

Και γι αυτό λαμβάνει την κατηγορία wikipedia-president.

iii. Person

Το PersonClassifier αναθέτει την κατηγορία wikibase-person χρησιμοποιώντας

κάποι α χαρκτηριστικά με την σειρά που περιγράφονται:

A. Category regexes

Χρησιμοποιεί τις ακόλουθες εκφράσεις για να ταυτίσει τις κατηγορίες

ενός άρθρου.

• .\* person

• ^\d+ deaths.\*

• ^\d+ births.\*

• .\* actors

• .\* deities

• .\* gods

• .\* goddesses

• .\* musicians

• .\* players

• .\* singers

B. Category regex excludes

Αποκλείει τις ακόλουθες regexes.

• \sbased on\s

• \sabout\s

• lists of\s

• animal\

C. Category matches

Γνωρίζουμε ότι ένα άρθρο αναφέρεται σε ένα πρόσωπο εάν η σελίδα ανήκει σε μια από τις ακόλουθες

mediawikiκατηγορίες:

• american actors

• american television actor stubs

• american television actors

• architects

• british mps

• character actors

• computer scientist

• dead people rumoured to be living

• deities

• disappeared people

• fictional characters

• film actors

• living people

• musician stubs

• singer stubs

• star stubs

• united kingdom writer stubs

• united states singer stubs

• writer stubs

• year of birth missing

• year of death missing

Για παράδειγμα η σελίδα του Leonardo DiCaprio έχει τις ακόλουθες κατηγορίες:

• Leonardo DiCaprio

• 1974 births

• Living people

• 20th-century American male actors

• 21st-century American male actors

• American environmentalists

• American film producers

• American male child actors

• American male film actors

• American male soap opera actors

• American male television actors

• American people of German descent

• American people of Italian descent

• American people of Russian descent

• American philanthropists

• Best Actor AACTA Award winners

• Best Actor Academy Award winners

• Best Drama Actor Golden Globe (film) winners

• Best Musical or Comedy Actor Golden Globe (film)

winners

• California Democrats

• Film producers from California

• Formula E team owners

• Male actors from Hollywood, California

• Male actors from Palm Springs, California

• Male actors of Italian descent

• People from Echo Park, Los Angeles

• Silver Bear for Best Actor winners

Όπως είναι φανερό η λίστα με τις κατηγορίες είναι αθυαίρετη και όχι πλήρης. Πολλαπλές μέθοδοι έχουν χρησιμοποιηθεί για να διορθωθεί αυτό.

Μερικές από αυτές είναι:

• Μέθοδοι με Supervised machine learning όπως SVM χρησιμοποιώντας άλλες μεθόδους να ορίσουν ένα πρόσωπο και να δημιουργήσουν εκπαιδευτικές ομάδες

• Hand-pick κοινές κατηγορίες για άρθρα προσώπων ορισμένα ξανά με διαφορετικά κριτήρια

(d) Resolvers

Οι Resolvers είναι επίσης μονήρεις (singletons) αλλά ο σκοπός τους είναι να βρούνε την αξία από την αναζητούμενη ιδιοκτησία (property)

. Όλοι οι resolvers προέρχονται από το BaseResolver

και πρέπει να εφαρμόζουν τις ακόλουθες μεθόδους:

• resolve(class, symbol, attribute): get the value of the

attribute of symbol symbol as class

• attributes(class, symbol): get a list of the attributes this

resolver can resolve.

Οι implemented resolvers είναι οι ακόλουθοι:

Error the minimum priority resolver, it will always resolve to an

error.

Infobox Resolve attributes found on infoboxes of a symbol.

Ένα πρόσωπο resolve τα ακόλουθα ειδικά χαρακτηριστικά των συμβόλων αναφερόμενα σε ανθρώπους :

• birth-date

• death-date

• gender

Κεφάλαια resolve το περιεχόμενοτων κεφαλαίων σε ένα άρθρο.

Term Can resolve a fixed set of ad-hoc attributes:

• coordinates *The coordinates of a geographical location*

• image *The image in the infobox*

• number *True if the symbol is plural (eg The Beatles)*

• proper *True if it refers to a unique entity.*

• short-article *A summary of the article. Typically the*

*first paragraph*

• url *The article url*

• word-cout *The size of the article*

(e) Lisp types

Ο τύπος Lisp είναι περιτυλίγματα (wrappers ) για python αντικείμενα ή τιμές που παρουσιάζονται σε μορφή s-expression που το START μπορεί να κατανοήσει . Έχουν δημιουργηθεί είτε από το ανεπεξέργαστο ερώτημα και έχουν ξετυλιχθεί (unwrapped) ώστε να είναι χρήσιμα στον αγωγό (pipeline ) , ή από την απάντηση που δίνει η WikipediaBase και στη συνέχεια κωδικοποιούνται σε ένα string και αποστέλλονται μέσω telnet στο START.

Chapter 6

Provider/Acquirer model

Η WikipediaBase προσπαθεί να είναι modular και με δυνατότητα επέκτασης . Για να επιτευχθεί αυτό,

Συχνά είναι χρήσιμο να συμπλέκει πολλαπλές πηγές του ίδιου τύπου του πόρου δεδομένων.

Αυτό είναι ιδιαίτερα χρήσιμο κατά την πρόσβαση heuristic μεθόδους όπως των classifier . για την

προώθηση της modularity και για νανα αποφευχθεί η σκληρή εξάρτηση δημιουργήθηκε το μοντέλο provider/acquirer:

Ο Provider είναι ένα αντικείμενο μέσω του οποίου μπορούμε να διαχειριστούμε πηγές που είναι αποθηκευμένες in a key-value fashion

Η κατηγορία (class) Provider προσφέρει διακοσμητές για να κάνει αυτή της διάταξη εύκολη. Ένας Acquirer έχει διαφανή (transparent) πρόσβαση στους πόρους πολλαπλών Provider s σαν να ήταν μια ενιαία τιμή. Αυτό το πρότυπο κυρίως χρησιμοποιείται για την KnowledgeBase ώστε να παρέχει το Frontend με τρόπο πρόσβασης στις πηγές.

1. TODO Παράδειγμα

Εκθέτουμε το μοτίβο με ένα παράδειγμα:

we will embed a small

lisp to python

from wikipediabase . provider import Provider ,

Acquirer , provide

class EvalContext ( Acquirer ):

def \_\_init\_\_ (self , closures ):

super ( EvalContext , self ). \_\_init\_\_ (

closures )

self . closures = closures

def \_\_call\_\_ (self , \_ctx , expr ):

if isinstance (expr , list ):

30

*# Handle quotes*

if expr [0] is ’quote ’:

return expr [1]

*# Call the lambda*

fn = self (\_ctx , expr [0])

return fn(self , \*[ self (\_ctx , e) for e

in expr [1:]])

if isinstance (expr , basestring ) and expr

in self . resources ():

return self (\_ctx , self . resources ()[

expr ])

return expr

class Lambda ( Acquirer ):

def \_\_init\_\_ (self , args , expr , env):

*# Get your symbols from all the available*

*closures plus an*

*# extra for local variables*

super (Lambda , self ). \_\_init\_\_ ([ env] + [

Symbols () ])

self . args = args

self . expr = expr

def \_\_call\_\_ (self , \_ctx , \* args ):

*# Add another closure to the list*

arg\_provider = Provider ();

for s, v in zip( self .args , args ):

arg\_provider . provide (s, v)

*# Build an eval context and run it*

ctx = EvalContext ([ arg\_provider , Provider

( self . resources ())])

return [ctx (ctx , e) for e in self . expr

][ -1]

class Symbols ( Provider ):

@provide (’setq ’)

def setq (self , ctx , symbol , val):

self . provide (symbol , val)

class Builtins ( Provider ):

31

@provide (’lambda ’)

def \_lambda (self , ctx , args , \* body ):

return Lambda (args , list ( body ), Provider (

ctx . resources ()))

@provide (’if ’)

def \_if(self , ctx , proposition , then , \_else ):

if ctx(ctx , proposition ):

return ctx (ctx , then )

else :

return ctx (ctx , \_else )

GLOBAL\_EVAL = EvalContext ([ Builtins () , Symbols ()

])

Αυτή η little lisp υποστηρίζει::

• lambdas

• A global symbol table

• lexical scoping

• conditionals

• Quoted literals

Είναι πράγματι πολυ μακριά από το να επιτυγχάνει remotely close tσε μια χρησιμοποιημένη γλώσσα αλλά μπορεί να πετύχει μερικά ενδιαφέροντα tricks:

Μπορούμε να εκτιμήσουμε python types:

>>> GLOBAL\_EVAL ({} , 1)

1

>>> GLOBAL\_EVAL ({} , True )

True

>>> GLOBAL\_EVAL ({} , " hello ")

’hello ’

>>> GLOBAL\_EVAL ({} , list )

<type ’list ’>

Μπορούμε να ορίσουμε lambdas και call them. Το ακόλουθο είναι ισοδύναμο με το

(*\_a:a*)1, το οπίο πρέπει να εκτιμηθεί στο 1:

>>> GLOBAL\_EVAL ({} , [[" lambda ", [’quote ’, [’a’]],

’a’], 1])

32

1

Η little lisp δεν είναι pure s επειδή έχουμε ένα global symbol table. Ο καλύτερος τρόπος να κάνουμε sequence expressions είναι να τα πακετάρουμε όλα σε ένα a lambda

και στη συνέχεια να τα εκτιμήσουμε :

>>> GLOBAL\_EVAL ({} , [[ ’lambda ’, [’quote ’, []] , [’

setq ’, ’b’, 2], ’b’]])

2

Ο προσεχτικός αναγνώστης ίσως παρατήρησε ότι η quoted list για ταlambda ar-

guments. Ο λόγος γι αυτό είναι ότι δεν θέλουμε η λίστα να εκτιμηθεί.

Πίσω στην αρχική μας εργασία. Σε κάθε σημείο του κώδικα των At each point in the code of embedded lisp symbols derive meaning από πολλαπλές πηγές :

• The local closure

• The arguments of the lambda

• Builtin functions

Όλα τα προηγούμενα περιληπτικοποιούνται χρησιμοποιώντας το provider-aquirer model.

Σε κάθε σημείο ένα διαφορετικό EvaluationContext είναι υπεύθυνο για την εκτίμηση και κάθε EvaluationContext έχει πρόσβαση στα γνωστά σύμβολα του μέσω μιας array

of providers τα οποία περιληπτικοποιούνται χρησιμοποιώντας το συζητημένο model.

33

Chapter 7

Testing

1. Unit testing

Η καλή λειτουργία της WikipediaBase εξασφαλίζεται από μια ολοκληρωμένη σειρά δοκιμών των unit tests, functional tests και regression tests.

(a) Unit tests

Τα Unit tests ελέγχουν μια μικρή ομάδα του functionality, το οποίο έχει συντεθεί για την δημιουργία του όλου συστήματος.

Για το unit testing χρησιμοποιούμε την βιβλιοθήκη python’s

default testing. Κάθε τεστ είναι μια τάξη (class )της υποτάξης (subclasse)s

(b) Functional and regression tests

Τα Functional tests είναι γραμμένα από πριν, κατά τη διάρκεια ή λίγο μετά της δημιουργίας του συστήματος και διεκδικούν τη σωστή συνολική λειτουργία του συστήματος. Τα Regression tests είναι πολύ παρόμοια με τα to functional tests . Αποδεικνύουν ότι όταν βρεθεί ένα σφάλμα(bug)το διορθώνουν και επιβεβαιώνουν ότι δεν θα εμφανισθεί ξανά αργότερα.

Τα Functional και τα regression tests είναι τοποθετημένα στα tests/examples.py

2. TODO Παραδείγματα

Virtually όλα τα τεστ ξεκινάνα με το ακόλουθο snippet:

from \_\_future\_\_ import unicode\_literals

try:

import unittest2 as unittest

except ImportError :

import unittest

from wikipediabase import fetcher

Το παραπάνω είναι ειδικό για το the fetcher module. Όπως είναι προφανές χρησιμοποιούμε το unittest module από την βιβλιοθήκη python. Το testτο ίδιο έχει το ακόλουθοformat:

class TestFetcher ( unittest . TestCase ):

def setUp ( self ):

self . fetcher = fetcher . get\_fetcher ()

def test\_html ( self ):

html = self . fetcher . html\_source ("Led␣

Zeppelin ")

self . assertIn (" Jimmy ␣ Page ", html )

Η setUp μέθοδος runs πρίν από κάθε τεστ του νTestCase. Τα τεστ του

testcase αντιπροσωπαε΄υονται από μεθόδους της class το οπίων το όνομα αρχίζει με τοwhose test\_. Στην συγκεκριμένη περίπτωση παίρνουμε την σελίδα της wikipedia για το

Led Zeppelin ότι το όνομα Jimmy Page αναφέρται τουλάχιστον μια φορά. Αυτό φανερά δεν συνάδει ότι το fetcher δεν φέρνει για παράδειγμα την σελίδα για το Yardbirds, Page’s first band. Γαι αυτό το λόγο γράφουμε μερικά από αυτού του είδους τεστ. Στην περίπτωση του

fetcher, για να ακολουθήσουμε το παράδειγμα , το συνολικό τεστ είναι:

class TestFetcher ( unittest . TestCase ):

def setUp ( self ):

self . fetcher = fetcher . get\_fetcher ()

def test\_html ( self ):

html = self . fetcher . html\_source ("Led␣

Zeppelin ")

self . assertIn (" Jimmy ␣ Page ", html )

def test\_markup\_source ( self ):

src = self . fetcher . markup\_source ("Led␣

Zeppelin ")

self . assertIn ("{{ Infobox ␣ musical ␣ artist ",

src)

def test\_unicode\_html ( self ):

35

html = self . fetcher . html\_source (u" Rhone ")

self . assertIn (" France ", html )

def test\_unicode\_source ( self ):

src = self . fetcher . markup\_source (" Rhone ")

self . assertIn (" Geobox | River ", src)

def test\_silent\_redirect ( self ):

*# redirects are only supported when*

*force\_live is set to True*

src = self . fetcher . markup\_source (" Obama ",

force\_live = True )

self . assertFalse (re. match ( fetcher .

REDIRECT\_REGEX , src))

Γράψαμε πολλαπλά τέτοια τέστ για να εξετάσουμε κάθε τμήμα της WikipediaBase.

3. Running tests

Εφαρμόσαμε το εργαλείο nosetests να βρούμε και να τρέξουμε τα τεστ. Για να το κάνουμε αυτό προσθέσαμε στο τεστ ένα προαπαιτούμενο στο setup.py και assign nose.collector to

manage our test suite:

from setuptools import setup

setup (

tests\_require =[

’nose >=1.0 ’,

...

],

...

test\_suite =’nose . collector ’,

...

)

TΣτη συνέχει να τρέξουμε τα τεστ:

$ python setup .py test

Η Nose θα βρει όκα τα αρχεία τα οποία είναι στα tests/ και έχουν τ prefix test\_,

για παράδειγμα test\_fetcher.py. Μέσα σ αυτά τα αρχεία η nose στα classes

τα subclass TestCase και των οποίων το όνομα αρχίζει με Test, για παράδειγμα

TestFetcher. Στη συνέχεια τρέχει όλες τις μεθόδους από τις collected classes που έχουν το test\_ prefix.

Είναι επίσης δυνατό να τρέξει ειδικά τεστ.

$ python setup .py test --help

Κοινές εντολές : ( βλέπε ’--help - commands ’ για περισσότερα )

setup .py build will build the package

underneath ’build /’

setup .py install will install the package

Global options :

-- verbose (-v) run verbosely ( default )

--quiet (-q) run quietly ( turns verbosity

off )

--dry -run (-n) don ’t actually do anything

--help (-h) show detailed help message

--no -user -cfg ignore pydistutils .cfg in your

home directory

Options for ’test ’ command :

--test - module (-m) Run ’test\_suite ’ in

specified module

--test - suite (-s) Test suite to run (e.g. ’

some\_module . test\_suite ’)

--test - runner (-r) Test runner to use

usage : setup .py [ global\_opts ] cmd1 [ cmd1\_opts ] [

cmd2 [ cmd2\_opts ] ...]

or: setup .py --help [ cmd1 cmd2 ...]

or: setup .py --help - commands

or: setup .py cmd --help

Για παράδειγμα:

$ python setup .py test -s tests . test\_lispify

running test

running egg\_info

writing requirements to wikipediabase .egg - info /

requires . txt

writing wikipediabase .egg - info /PKG - INFO

writing top - level names to wikipediabase .egg - info

/ top\_level .txt

writing dependency\_links to wikipediabase .egg -

info / dependency\_links .txt

writing entry points to wikipediabase .egg - info /

entry\_points . txt

reading manifest file ’ wikipediabase .egg - info /

SOURCES .txt ’

reading manifest template ’MANIFEST .in ’

writing manifest file ’ wikipediabase .egg - info /

SOURCES .txt ’

running build\_ext

test\_bool ( tests . test\_lispify . TestLispify ) ... ok

test\_bool\_with\_typecode ( tests . test\_lispify .

TestLispify ) ... ok

test\_date\_multiple\_voting ( tests . test\_lispify .

TestLispify ) ... ok

test\_date\_simple ( tests . test\_lispify . TestLispify )

... ok

test\_date\_with\_range ( tests . test\_lispify .

TestLispify ) ... ok

test\_dict ( tests . test\_lispify . TestLispify ) ... ok

test\_dict\_with\_escaped\_string ( tests . test\_lispify

. TestLispify ) ... ok

test\_dict\_with\_list ( tests . test\_lispify .

TestLispify ) ... ok

test\_double\_nested\_list ( tests . test\_lispify .

TestLispify ) ... ok

test\_error ( tests . test\_lispify . TestLispify ) ...

ok

test\_error\_from\_exception ( tests . test\_lispify .

TestLispify ) ... ok

test\_keyword ( tests . test\_lispify . TestLispify ) ...

ok

test\_keyword\_with\_typecode ( tests . test\_lispify .

TestLispify ) ... ok

test\_list ( tests . test\_lispify . TestLispify ) ... ok

test\_list\_of\_dict ( tests . test\_lispify . TestLispify

) ... ok

test\_list\_of\_dict\_with\_typecode ( tests .

test\_lispify . TestLispify ) ... ok

test\_list\_with\_typecode ( tests . test\_lispify .

TestLispify ) ... ok

test\_nested\_list ( tests . test\_lispify . TestLispify )

... ok

test\_none ( tests . test\_lispify . TestLispify ) ... ok

test\_none\_with\_typecode ( tests . test\_lispify .

38

TestLispify ) ... ok

test\_number ( tests . test\_lispify . TestLispify ) ...

ok

test\_number\_with\_typecode ( tests . test\_lispify .

TestLispify ) ... ok

test\_string ( tests . test\_lispify . TestLispify ) ...

ok

test\_string\_escaped ( tests . test\_lispify .

TestLispify ) ... ok

test\_string\_not\_keyword ( tests . test\_lispify .

TestLispify ) ... ok

test\_string\_with\_typecode ( tests . test\_lispify .

TestLispify ) ... ok

test\_unicode\_string ( tests . test\_lispify .

TestLispify ) ... ok

----------------------------------------------------------------------

Τρέχει 27 τεστ σε 0.047 s

OK

Chapter 8

Synonyms

Πριν μιλήσουμε για τα συνώνυμα είναι σημαντικό να ορίσουμε τα σύμβολα στο πεδίο του

omnibase universe:

Σύμβολα είναι ταυτοποιητές των "αντικειμένων" "objects" στις πηγες των πληροφοριών ( ο όρος "συμβολο"("symbol")είναι ατυχής γιατί έχει διάφορες έννοιες στην επιστήμη των υπολογιστών. Δυστυχώς έχει μείνει για ιστορικούς λόγους.)

Δεδομένου ότι η γλώσσα τείνει να έχουν πολλαπλές λέξεις που αναφέρονται στο ίδιο πράγμα, Είναι επιτακτική η ανάγκη να καθορισθούν ονόματα για τα σύμβολα. Συνώνυμα είναι τα ονόματα τα οποία οι χρήστες μπορούν να χρησιμοποιήσουν για να αναφερθούν στα σύμβολα.

( Ο όρος συνώνυμα "synonym" είναι ατυχής γιατί είναι one-way mapping -"gloss"θα ήταν καλύτερος όρος αλλά έμεινε ο όρος συνώνυμα για ιστορικούς λόγους)

Ο όρος συνώνυμα είναι η ίδια η δουλειά του backend. Για το λόγο αυτό αναλαμβάνει η Therefore it

WikipediaBase να ορίσει τα απαιτούμενα συνώνυμα.

1. Good/Bad συνώνυμα

Υπάρχουν κανόνες για το ποιο είναι καλό ή κακό συνώνυμο

In short synonyms:

• Should not lead with articles ("the", "a", "an")

• Should not lead with "File:" or "TimedText:".

• Should not fragment anchors. Eg "AlexanderPushkin#Legacy"

• Should not start with the following:

40

– "List of "

– "Lists of "

– "Wikipedia: "

– "Category: "

– ":Category: "

– "User: "

– "Image: "

– "Media: "

– "Arbitration in location"

– "Communications in location"

– "Constitutional history of location"

– "Economy of location"

– "Demographics of location"

– "Foreign relations of location"

– "Geography of location"

– "History of location"

– "Military of location"

– "Politics of location"

– "Transport in location"

– "Outline of topic"

• Δεν πρέπει να ταιριάζει \d\d\d\d στο location ή location στο \d\d\d\d

• Δεν πρέπει να είναι ονόματατων disabiguation pages. Για ναto κάνουμε αυτό έτσι ώστε να συμπεριλαμβάνει όλες τις σχετικές σελίδες , συμπεριλαμβανομένων των typos, αυτό σημαίνει σύμβολα που ταιριάζουν με \([Dd]isambig[^)]\*\)

•Συνώνυμα που ότι τόσο α) θα μπορούσαν να εκληφθούν ότι ξεκινούν με άρθρα και β ) μπορεί να υποτάσσουν κάτι χρήσιμο . Αυτό σημαίνει ότι για παράδειγμα « A. House» ( συνώνυμο του « Abraham House " ) είναι ελλειπών προδιαγραφών (dies-

qualified) διότι ενδέχεται να παραπλανήσει START στην περίπτωση των ερωτήσεων όπως « Πόσο κοστίζει ένα σπίτι στη Silicon Valley ; » . Αφετέρου "a priori " μπορεί να διατηρηθεί επειδή δεν υπάρχουν λογικές ερωτήματα όπου "α" είναι ένα άρθρο πριν " priori " .

2. Synonym generation

Για να διαλευκάνουμε αυτούς τους περιορισμούς δύο μέθοδοι χρησιμοποιούνται . To accommodate these restrictions two methods are employed. Dis-

qualification και modification των υποψήφιων συνονύμων. Πρώτα προσπαθούμε τη modification και αν αυτό αποτύχει επιχειρούμε να κάνουμε disqualify. Οι κανόνες για modification έχουν ως εξής

• Να κάνουμε Strip τα determiners (articles) που είναι στην αρχή από ένα συνώνυμο

(ή θα μπορούσε να ήταν στην αρχή εάν όχι γιαpunctuation):

– "A "

– "An "

– "The "

– ’(The) ’

– The&nbsp;

– etc.

• Generate και τα δύο versions, με και χωρίς paren. Eg given symbol

"Raven (journal)" generate both:

– "Raven (journal)"

– "Raven"

• Generate πριν και μετά τοslash, αλλά όχι το αρχικόsymbol, e.g.:

– Given symbol "Russian language/Russian alphabet" generate

both

\* "Russian language"

\* "Russian alphabet"

• Reverse inverted synonyms με commas. Eg given synonym

"Congo, Democratic Republic Of The" invert it to get "Demo-

cratic Republic Of The Congo"

•Ώς συνήθως , απέρριψε leading articles εάν είναι αναγκαίο. Π.χ. δοθέντος synonym

"Golden ratio, το" replace it with "the Golden ratio", στη συνέχειΑ strip

articles για να πάρεις: "Golden ratio" το ίδιο συμβάνει για τα a, an, etc.

Με αυτό τον τρόπο κάναμε generate ένα αρχικό πακέτο συνωνύμων από το ίδιο το όνομα του αντικειμένου

. Επιπλέον μπορούμε να κάνουμε generate ένα πακέτο από από τα

wikipedia redirects στο άρθρο. Η Wikipedia παρέχει ένα SQL

dump για όλα τα redirects.

Για να φορτώσουμε τον πίνακα στην βάση δεδομένων όπου έχουμε φορτώσει τα δεδομένε α της wikipedia, πρέπει να φορτώσουμε τον πίνακα των redirects:

wget https :// dumps . wikimedia . org/ enwiki / latest /

enwiki - latest - redirect .sql.gz \

-O redirect .sql .gz && gzcat redirect .sql.gz |

mysql

42

Και στη συνέχεια το SQL db για να βρούμε όλα τα συνώνυμα του (καλά και κακά)

Bill Clinton μπορούμε να:

mysql > select page\_title , rd\_title from redirect

join page on rd\_from = page\_id and ( rd\_title =

" Bill\_Clinton " or page\_title = " Bill\_Clinton "

);

+ *--*

*-----------------------------------+--------------+*

| page\_title | rd\_title

|

+ *--*

*-----------------------------------+--------------+*

| BillClinton |

Bill\_Clinton |

| William\_Jefferson\_Clinton |

Bill\_Clinton |

| President\_Clinton |

Bill\_Clinton |

| William\_Jefferson\_Blythe\_IV |

Bill\_Clinton |

| Bill\_Blythe\_IV |

Bill\_Clinton |

| Clinton\_Gore\_Administration |

Bill\_Clinton |

| Buddy\_ ( Clinton ’s\_dog )␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ Bill\_clinton ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ William\_Jefferson\_Blythe\_III ␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ President\_Bill\_Clinton ␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ Bull\_Clinton ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ Clinton , \_Bill ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ William\_clinton ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣42 nd\_President\_of\_the\_United\_States ␣|␣

Bill\_Clinton ␣|

|␣ Bill\_Jefferson\_Clinton ␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

43

|␣ William\_J . \_Clinton ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ Billl\_Clinton ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ Bill\_Clinton \␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ Bill\_Clinton ’ s\_Post\_Presidency |

Bill\_Clinton |

| Bill\_Clinton ’s\_Post - Presidency ␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣Klin -ton␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ Bill\_J . \_Clinton ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ William\_Jefferson\_ " Bill " \_Clinton ␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ William\_Blythe\_III ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ William\_J . \_Blythe ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ William\_J . \_Blythe\_III ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ Bil\_Clinton ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ WilliamJeffersonClinton ␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ William\_J\_Clinton ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣

Bill\_Clinton ␣|

|␣ Bill\_Clinton ’ s\_sex\_scandals |

Bill\_Clinton |

| Billy\_Clinton |

Bill\_Clinton |

| Willam\_Jefferson\_Blythe\_III |

Bill\_Clinton |

| William\_ " Bill " \_Clinton |

Bill\_Clinton |

| Billll\_Clinton |

Bill\_Clinton |

| Bill\_Klinton |

Bill\_Clinton |

| William\_Clinton |

Bill\_Clinton |

| Willy\_Clinton |

Bill\_Clinton |

| William\_Jefferson\_ ( Bill ) \_Clinton |

44

Bill\_Clinton |

| Bubba\_Clinton |

Bill\_Clinton |

| MTV\_president |

Bill\_Clinton |

| MTV\_President |

Bill\_Clinton |

| The\_MTV\_President |

Bill\_Clinton |

| Howard\_G . \_Paster |

Bill\_Clinton |

| Clintonesque |

Bill\_Clinton |

| William\_Clinton |

Bill\_Clinton |

| William\_Jefferson\_Clinton |

Bill\_Clinton |

+ *--*

*-----------------------------------+--------------+*

46 rows in set (11.77 sec)

45

Chapter 9

Backend databases

Η Wikipediabase χρησιμοποιεί κυρίως έναν απομακρυσμένο χώρο αποθήκευσης δεδομένων και εφαρμόζει το mediawiki interface. Προσπαθεί να αντιμετωπίσει ζητήματα επιδόσεων που προκύπτουν με την προσωρινή αποθήκευση των σελίδων σε μια βάση δεδομένων backend key-value. Το interface με τη βάση δεδομένων is abstracted με τη χρήση ενός λεξικού python - -style interface, το οποίο εφαρμόζεται στο persistentkv.py .Εφαρμογές των back-ends παρουσιάζονται παρακάτω, αλλά είναι ασήμαντο να παρέχεται κάθε backend που ο καθένας συναντά.

Ένα άλλο χαρακτηριστικό που το interface στην βάση δεδομένων πρέπει να μπορεί να χρησιμοποιήσει είναι η κωδικοποίηση των αποθηκευμένων αντικειμένων. Επειδή όλη η αποθηκευμένη πηροφορία είναι κείμενο, η βάση δεδομένων πρέπει ν είναι ικανή να ανασύρει ακριβώς το κείμενο που έχει αποθηκευθεί λαμβάνοντας υπόψη την κωδικοποίηση.

Λόγω των περιορισμών του DBM’s τα κλειδιά( keys ) πρέπει να είναι μόνο κωδικοποιημένα ASCII η βασική τάξη (class) για interfacing με την βάση δεδομένων. Το EncodedDict, εφρμόζει τις μεθόδους \_encode\_key και\_decode\_key (that default να ταυτοιποιήσει λειτουργίες) για να παραχωρήσει ένα εύκολο hook εφαρμογών με σκοπό να διαχειρισθεί πιθανές καταστάσεις

1. DBM

Διάφορα dbm εφαρμογές παρέχονται από την σταθερή βιβλιοθήκη της python

Όμως καμιά από τις εφαρμογές που shipped with python δεν είναι μέρος της σταθερής βιβλιοθήκης της python. Μερικές εφαρμογές DBM που είναι διαθέσιμες μέσω της σταθερής βιβλιοθήκης της python είναι:

• AnyDBM

• GNU DBM

• Berkeley DBM

Είναι σημαντικό να αναφέρουμε ότι η ομαλή λειτουργία αθτών των βιβλιοθηκών εξαρτάται σε σημαντικό βαθμό από την βασική πλατφόρμα. Όπως αναφέρθηκε παραπάνω

the interface classes to DBM transcode keys to ASCII. Ο ακριβής μηχανισμός που γίνεται αυτό είναι:

def \_encode\_key (self , key):

if isinstance (key , unicode ):

return key . encode (’ unicode\_escape ’)

return str (key )

def \_decode\_key (self , key):

return key . decode (’ unicode\_escape ’)

2. SQLite

Το SQLite επίσης θεωρείται ως caching backend βάση δεδομένων. Δυστυχώς η αποτελεσματικότητά του στο δικό μας σκοπό ήταν απογοητευτική.

Χρησιμοποιήσαμε ένα very thin wrapper, sqlitedict, για να πάρουμε ένα key-value interface to SQLite – μια relationalβάση δεδομένων. Ο σχετικός WikipediaBase κώδικας είναι πολύ σύντομος ( short):

from sqlitedict import SqliteDict

class SqlitePersistentDict ( EncodedDict ):

def \_\_init\_\_ (self , filename , configuration =

configuration ):

if not filename . endswith (’. sqlite ’):

filename += ’. sqlite ’

db = SqliteDict ( filename )

super ( SqlitePersistentDict , self ).

\_\_init\_\_ (db)

def sync ( self ):

self .db. close ()

super ( SqlitePersistentDict , self ). sync ()

Below are two benchmark functions that will read/write 100000 times

to a key-value database.

47

def benchmark\_write (dic , times =100000) :

for i in xrange ( times ):

dic[’o’ + str (i)] = str(i) \* 1000

def benchmark\_read (dic , times =100000) :

for i in xrange ( times ):

dic[’o’ + str (i)]

And here they are run over memory based tmpfs on deban.

>>> import timeit

>>> sqlkv = SqlitePersistentDict (’/tmp/ bench1 .

sqlite ’)

>>> timeit . timeit ( lambda : benchmark\_write ( sqlkv )

, number =100)

10.847157955169678

>>> timeit . timeit ( lambda : benchmark\_read ( sqlkv ),

number =100)

18.88098978996277

>>> dbmkv = DbmPersistentDict (’/tmp/ bench .dbm ’)

>>> timeit . timeit ( lambda : benchmark\_write ( dbmkv )

, number =100)

0.18030309677124023

>>> timeit . timeit ( lambda : benchmark\_read ( dbmkv ),

number =100)

0.14914202690124512

Η DBM βαση δεδομένων είναι σχεδόν 10 φορές ταχύτερη από sqlite. Η διαφορά στην εκτέλεση οφείλεται στις διαφορετικές committing policies που έχουν μεταξύ τους.

Μπορεί να είναι δυνατόν να ρυθμιστεί το SQLite ώστε να είναι τόσο γρήγορο όσο η DBM αλλά όχι με κάποιον εύκολο τρόπο.

3. Άλλα backends

Άλλα backends λαμβάνονται υπόψη, κυρίως το Redis το οποίο εφαρμόσθηκε αμέσως μετά την παράδοση της εργασίας από τον Alvaro Morales.

Ο λόγος που αρχικά δεν το χρησιμοποιήσαμε ήταν γιατί έχει μοντελοποιηθεί ως ένας server-client και προσθέτει περιπλοκότητα σε ένα τμήμα του συστήματος το οποίο πρέπει να είναι όσο το δυνατόν πιο απλό. Ένας άλλος λόγος του αρχικού προβληματισμού μας ήταν σχετικά με το third partyπ.χ. ie. not shipped with python – databases ήταν να αποφευχθούν επιπλέον εξαρτήσεις ειδικά όταν είναι the cool database du jour.

Chapter 10

Data sources

1. HTML and MediaWiki API

Η αρχική προσέγγιση για να πάρουμε τα δεδομένα είναι να ανασύρουμε τις φυσιολογικές HTML εκδόσεις των άρθρων της wikipedia και χρησιμοποιώντας edit pages να ανασύρουμε το mediawiki markup. Ανεξαιρέτως χρησιμοποιήσαμε το αρχικό wikipedia.org site για λόγους εκτέλεσης (performance ) (Βλέπε κεφάλαιο wikipedia-mirror runtime performance sec-

tion).

Το Mediawiki παρέχει a RESTful API για όλη την απαιτούμενη λειτουργία (functionality).

Η βαική αρχή είναι ότι κάποιος μπορεί να στείλει αιτήματα με μεθόδους POST ή GET

και να λαμβάνει απάντηση με την μορφή XML ή JSON. Η προτιμητέα απάντηση για την

WikipediaBase ήταν ν α στέλνονται GET HTTP αιτήματα και να λαμβάνουν JSON δεδομένα. Το GET επιλέχθηκε ειδικά προτάθηκε στην mediawiki API page γιατί caching συμβαίνει στο HTTP επίπεδο.

Σύμφωνα με τις οδηγίες του HTTP τα POST αιτήματα δεν μπορούν να cached.

Για το λόγο αυτό όταν διαβάζει κάποιος δεδομένα από web service API, θα πρέπει να χρησιμοποιεί GET αιτήματα και όχι POST.

Επίσης πρέπει να σημειωθεί ότι ένα αίτημα δεν μπορεί να εκτελεσθεί από cache εκτός αν το

URL είναι ακριβώς το ίδιο.

Εαν ζητήσεις ένα αίτημα για

api.php?. . . .titles=Foo|Bar|Hello, and cache the result, then

a request for api.php?. . . .titles=Hello|Bar|Hello|Foo will not

go through the cache even though MediaWiki returns the

same data!

JSON was selected simply because the python json package in the stan-

dard library is much easier to use than lxml, the library we use for

XML/HTML parsing.

49

2. Dumps / Database

Direct interface με μια τοπική βάση δεδομένων, εκτός από caching χρησιμοποιώντας mdb and/or sqlite δεν συμπεριλαμβάνεται σαν μέρος της παρούσας εργασίας. Όμως shotrly after caching and compile time data pools in redis and postrgres were

implemented.

50

Chapter 11

Date parser

Dateparser υπάρχει σε ένα ξεχωριστό πακέτο που ονομάζεται overlay-parse

1. Parsing with overlays

Η έννοια του overlay εμπνεύστηκε από τα emacs overlays. Είναι αντικείμενα (objects) που εξειδικεύουν την συμπεριφορά ενός υποσυνόλου του κειμένου με το να του δίνουν ιδιότητες για παράδειγμα το κανουν clickable ή highlighted.

An overlay over part of text *t* in our context is

• a tuple representing the range within that text

• a set of tags that define semantic sets that the said substring is a

member of

• arbitrary information (of type *A*) that the underlying text de-

scribes.

More formally:

*oi 2 TextRanget \_Set*(*Tag*) *\_ A numbers*

*Text ! fo*1*; o*2*; :::; ong*

So for example out of the text

*Theweather today;*

z }*o*1| {

*Tuesday*

z}*o*2|{

21*st of*

z }*o*3| {

*November*

z}*o*4|{

2016*; was sunny:*

51

We can extract overlays *fo*1*; :::; o*4*g*, so that

*o*1 = ( *r*("*Tuesday*")*; f*DayOfWeek*;* FullName*g;* 2)

*o*2 = ( *r*("21*st*")*; f*DayOfMonth*;* Numeric*g;* 21)

*o*3 = ( *r*("*November*")*; f*Month*;* FullName*g;* 11)

*o*4 = ( *r*("2016")*; f*Year*;* 4digit*g;* 2016)

Παρατηρείστε ότι όλα τα overlays του παραδείγματος έχουν *A* = N, όπως κωδικοποιούμε την ημέρα της εβδομάδος, τη μέρα του μήνα, το μήνα του έτους ως φυσικούς αριθμούς.

Κωδικοποιούμε πιο ακριβή πληροφορία (πχ αυτή η μέρα είναι διαφορετική από την μήνα από την φύση της) στοtag set.

Μόλις έχουμε ένα σύνολο από overlays μπορούμε να ορίσουμε μια overlay sequences ως

overlays τα οποία έχουν συνεχόμενο εύρος, Αυτά και τα δικά τους tag sets ταυτίζονται με ειδικά μοτίβα . Για παράδιγμα μπορούμε να ψαξουμε για σειρές από overlays που ταυτίζονται με το μοτίβο.

*p* = DayOfMonth*;* Separator(*=*)*;* (Month*^*Number)*;* Separator(*=*)*;*Year

to match patterns like 22*=*07*=*1991, where *Separator*(*=*) matches only

the character "/"

2. The dates example

The working example and motivation of the package is date parsing.

The dates submodule is itself about 200 lines of code exposes two main

entry points:

• just\_dates that looks for dates in a text.

• just\_ranges that looks for data ranges in a corpus.

Below are presented some examples. Note that 0 means unspecified

>>> from overlay\_parse . dates import just\_dates ,

just\_ranges , just\_props

>>> just\_dates (" Timestamp :␣ 22071991: ␣She␣ said ␣she

␣was ␣\

␣␣␣␣␣␣␣␣ coming ␣on␣ april ␣the ␣18th ,␣it ’s␣26␣ apr␣

2014 ␣ and␣ hope ␣is␣ leaving ␣me.")

... [(22 , 7, 1991) , (18 , 4, 0) , (26 , 4, 2014) ]

52

>>> dates = just\_dates ("200␣AD␣300␣b.c.")

>>> just\_dates ("200 ␣AD␣ 300␣b.c.")

[(0 , 0, 200) , (0, 0, -300)]

>>> just\_ranges (u"I␣ will ␣be␣ there ␣ from ␣ 2008 ␣to␣

2009 ")

[((0 , 0, 2008) , (0, 0, 2009) )]

>>> just\_ranges ("I␣ will ␣ stay ␣ from ␣ July ␣the␣20 th␣

until ␣ today ")

[((20 , 7, 0) , (29 , 4, 2016) )]

>>> just\_dates (’{{ Birth ␣ date ␣ and␣ age |1969|7|10| df

=y}} ’)

[(10 , 7, 1969) ]

>>> just\_ranges (u’German :␣[\ u02c8v \ u0254lf \ u0261a

\ u014b ␣ama \ u02c8de \ u02d0 \ u028as ␣\ u02c8mo \

u02d0tsa \ u0281t ],␣ English ␣see␣fn .;[1] ␣27␣

January ␣ 1756\ xa0\ u2013 ␣5␣ December ␣ 1791 ’)

[((27 , 1, 1756) , (5, 12, 1791) )]

53

Part IV

WikipediaMirror

54

Wikipedia mirror είναι ένα σύστημα με στόχο να αυτοματοποιήσει τη δημιουργία ενός τοπικού

κλώνος της wikipedia περιέχοντας μόνο τα άρθρα - δεν περιέχει τους χρήστες ,

συζήτηση και επεξεργασμένη ιστορία. Η αυτοματοποιημένη διαδικασία περιλαμβάνει τη ρύθμιση

ενός διακομιστή, μια βάση δεδομένων και γέμισμα αυτής της βάσης δεδομένων με τα άρθρα της wikipedia. Ο σκοπός για αυτό είναι να παρέχει την δυνατότητα πρόσβασης του συνόλου των δεδομένων της Wikipedia, ανεξάρτητα από το wikipedia.org .

Chapter 12

mediawiki stack overview

Wikipedia - mirror βασίζεται στο MediaWiki stack που παρέχεται από το Bitnami . Μιά

υπηρεσία που χτίζει το σύνολο του διακομιστή εντός των ορίων ενός καταλόγου. Αυτό

είναι χρήσιμο γιατί αποφεύγεται η επιβάρυνση της χρήσης container ή VM

τεχνολογίας και μας δίνει τη δυνατότητα να έχουμε άμεση πρόσβαση στο σύστημα αρχείων του stack, ενώ εξακολουθούσαμε να έχουν το σύστημα κατασκευής Bitnami να κάνει την κοπιώδη εργασία της ενορχήστρωσης των διαφόρων τμημάτων και επίσης διαχωρίζεται ο διακομιστής από το υπόλοιπο συστήματος.

Το stack αποτελείται από

• An http server, in our case apache

• The web application runtime, in our case PHP

• A database, in our cas MySQL

• The web application itself, in our case mediawiki

Όλα τα παραπάνω παρέχονται από το the bitnami mediawiki stack. Xampp

used to be go-to for that αλλά είναι unmaintained έτσι αποφασίσαμε να χρησιμοποιήσουμε το bitnami το οποίο δουλεύει αρκετά καλά.

Όταν το stack είναι ρυθμισμένο κατάλληλα το wikipedia dump xmlκατεβάιναι

και μετατρέπεται σε sql dump μεmwdumper. Θα μπορούσε να piped άμεσα

στο MySQL? αλλά η εξαγωγή παίρνει χρόνο τα πράγματα μπορεί να χειροτερέψουν κατα το and

dumping.

1. Στοιχεία του stack.Παρουσιάζεται κάθε στοιχείο του stack με περισσότερες λεπτομέρειες παρακάτω.

(a) Apache

As per wikipedia:

56

The Apache HTTP Server, colloquially ονομάζεται Apache, είναι παγκοσμίως ο πιο χρησιμοποιημένος web server software. Αρχικά ήταν εγκατεστημένος στο Originally based on NCSA HTTPd server. Η ανάπτυξη του Apache άρχισε το στις αρχές του 1995 μετά από εργασίες στο

NCSA code stalled. Apache έπαιξε σημαντικό ρόλο στην αρχική ανάπτυξη του

World Wide Web. Γρήγορα   
προσπέρασε το NCSA HTTPd ως κυρίαρχο HTTP server, και έχει παραμείνει πιο δημοφιλής από τον Απρίλιο του 1996. Το 2009, έγινε το πρώτο λογισμικό web server και εξυπηρετεί περισσότερες από 100 εκατομμύρια ιστοσελίδες.

Apache αναπτύσσεται και συντηρείται από μια ανοιχτή κοινότητα προγραμματιστών υπό την αιγίδα του Apache Software Foundation. Συνήθως χρησιμοποιείται σε ένα σύστημα τύπου Unix (συνήθως Linux ). Το λογισμικό είναι διαθέσιμο για μια ευρεία ποικιλία λειτουργικών συστημάτων εκτός Unix, συμπεριλαμβανομένων eComStation , Microsoft Windows , NetWare , OpenVMS , OS / 2 , και TPF . Που διατίθεται βάσει της άδειας Apache . Το Apache είναι δωρεάν και open-source λογισμικό.

Είναι δίκαιο να πούμε ότι apache είναι ένα από ταους πιο δημοφιλείς

διακομιστές web στο διαδίκτυο. Η ίδια η wikipedia.org φαίνεται να χρησιμοποιεί

ένα πιο σύνθετο stack που περιλαμβάνει varnish, ένα HTTP επιταχυντή ,

και nginx , μια εναλλακτική λύση , επίσης αρκετά δημοφιλή διακομιστή HTTP .

Καταλήξαμε σε αυτό το συμπέρασμα από την επιθεώρηση των headers που επιστρέφονται από τη

wikipedia.org . Στην περίπτωση http://www.wikipedia.org ανακατευθυνόμαστε

προς το secure domain (προσοχή στο διακομιστή : line):

$ curl -s -D - http :// www. wikipedia .org -o /

dev / null

HTTP /1.1 301 TLS Redirect

Server : Varnish

[...]

And if we directly ask for https://www.wikipedia.org nginx

seems to be handling our request:

$ curl -s -D - https :// www. wikipedia .org -o /

dev / null

HTTP /1.1 200 OK

Server : nginx /1.9.4

[...]

57

Ωστόσο, είναι πέρα από το πεδίο της συγκεκριμένης εργασίας να αναπαράγουμε με ακρίβεια υποδομή της Wikipedia . Έχουμε επικεντρωθεί στην λειτουργικότητα. Σ αυτό λόγω της δημοτικότητας, και της εν δυνάμει ταχύτητας των ρυθμών της αυτόματης εγκατάστασης το Bitnami MediaWiki stack χρησιμοποιήθηκε ως διακομιστή μας

However it is beyond the scope of the project to precisely replicate

wikipedia’s infrastructure. We focus on the functionality. There-

fore due to the popularity, familiarity and by virtue of apace being

part of the automatically installable bitnami mediawiki stack, we

use it as our server.

(b) PHP

Η MediaWiki , η οποίο συζητείται αργότερα, είναι γραμμένη εξ ολοκλήρου σε PHP , μια

δημοφιλής πλευρά του server , με δυναμική δακτυλογράφηση, προσανατολισμένη στα αντικείμενα , γλώσσα scripting. Το PHP είναι απαραίτητο και είναι εγκατεστημένο με το Bitnami mediawiki stack. Το PHP είναι δημοφιλής ανάμεσα στους προγραμματιστές του web και αυτό οφείλεται εν μέρει στην υποστήριξη που έχει από πολλούς σχετικές βιβλιοθήκες με βάσεις δεδομένων (συμπεριλαμβανομένων

ING PostgreSQL , MySQL , Microsoft SQL Server και SQLite ) και

είναι ουσιαστικά ένα template δημιουργίας προτύπων γλώσσας HTML.

(c) MySQL

Mediawiki can use a number of different SQL database backends:

• MSSQL: An SQL database by Microsoft

• MySQL: Using the standard PHP library for MySQL.

• MySQLi: An extension to the MySQL backend

• Oracle: A propertiary SQL database by Oracle.

• SQLite: An SQL database that is typically accessed as a

library rather than over a client-server scheme as is the case

with the other options on the list.

Η Wikipedia παρέχει πολλαπλά dump files για τους SQL πίνακες δευτερογενούς σημασίας στο MySQL format (eg. page redirects, cate-

gories etc) και προτείνει mwdumper which parses the XML dumpls

των άρθρων της wikipedia στο MySQL. Αυτό και το παρεχόμενο bitnami ανήκει στοαυτοποιημένο της stack, κάνει το MySQL την προφανή επιλογήγια το wikipedia-mirror stack.

(d) MediaWiki

Mediawiki είναι η καρδιά της wikipedia.

58

Chapter 13

Setting up

Στη συν'εχεια είναι βήμα το βήμα οδηγείες First, clone το git repo:

$ git clone https :// github .com/ fakedrake / wikipedia -

mirror

$ cd wikipedia - mirror

σ αυτό το σημείο θεωρητικά κάποιος μπορεί να τρέξει make sql-load-dumps τα οποία θα φροντίσουν να stting up οτιδήποτε χρειάζεται να φορτωθείη βάση δεδομένων των dumps

σε μια λειτουργική SQL βάση δεδομένων. Φυσικά για να γίνει αυτό πρώτα θα πρέπει να εκτελεσθούν μερικά βήματα.

• Download the wikipedia database dumps in XML format.

• Transform them into a format that MySQL understands.

• Set up the bitnami stack that includes a local install of MySQL

• Load the MySQL dumps into MySQL

Όλα αυτά τα βήματα κωδικοποιούνται ως τμήμα μιας εξαρτώμενης ιεραρχίας κωδικοποιούμενα σε

makefile targets και στη θεωρία αυτό πραγματοποιείται αυτόματα και αποτελεσματικά οδηγείται σε αποτελεσματική wikipedia mirror. Όμως αυτή λειτουργία είναι μεγάλη και ευθραυστη και συνιστάται κάθε βήμα να γίνεται εξατομικευμένα και χειρονακτικά.

Πρώτα , κατεβάζουμε και εγκαθιστάμε το bitnami. Η ακόλουθη εντολή θα fetch

έναν executable από το bitnami website και θα κάνει μια τοπική εγκατάσταση του

bitnami stackόπως συζητήθηκε παραπάνω:

59

$ make bmw - install

Το επόμενο βήμα είναι να βεβαιωθούμε ότι το maven, η java είναι ένα is a software project management and comprehension είναι εγκαταστημένα, απαιτείται να εγκατασταθεί και να στηθεί το mwdumper (βλέπε παρακάτω). Μπορεί να γίνει αυτό αν βεβαιωθούμε ότι τα παρακάτω έχου επιτευχθεί.:

$ mvn -- version

Σημείωση: if running on Ubuntu 14.04, ίσως χρειασθεί να εγκαταστήσουμε τοMaven (για

Java) χρησιμοποιώντας sudo apt-get install maven.

Τώρα όλα έχουν εγκατασταθεί στο automatically download Wikipedia’s XML

dumps και στη συνέχεια τα μετατρέπει σε SQL χρησιμοποιώντας maven. Πρώτα το maven θα πρέπει να κατέβει και να χτισθεί.

Μετά τα compressed XML dumps θα πρέπει να κατέβουν από την wikipedia. Θα γίνουν uncompressed και τελικά θα μετατραπούν σ MySQL dumps χρησιμοποιώντας mwdumper. Αυτή είναι πολύ χρονοβόρα διαδικασία και χρειάζεται 6-11 ώρες σε ένα τυπικό μηχάνημα

$ make sql -dump - parts

Όταν αυτο γίνει επιτυχώς μπορούμε να φορτώσουμε τα SQL dumps στη βάση δεδομένων MySQL

$ make sql -load - parts

Τελικά τα

$ make mw - extensions

60

Chapter 14

Tools

Ένας αριθμός εργαλείων αναπτύχθηκε για την χρήση:

1. pageremover.c

Όπως προηγουμένως συζητήθηκε η xerces βιβλιοθήκη στην οποία τα mwdumper εξαρτώνται απέτυχε,φαινιομενικά τυχαία να επεξεργαστεί κάποιες σελίδες. Για να διευθετηθεί αυτό το πρόβλημα αφαιρέσαμε τις σελίδες πλήρως και ξαναπροσπαθήσαμε.

Επειδή αυτή η εργασία είναι εύκολη και η λειτουργία ευαίσθητη γράψαμε ένα χαμηλού επιπέδου πρόγραμμα στη C για να το επιλύσουμε.

page\_remove.c. Το Pageremover δέχεται ως input the path του XML wikipedia dump, the

offset του άρθρου και το μέγεθος του άρθρου.

Μετά χρησιμοποιεί το mmapsystem call στο random-access τα δεδομένα μέσα στο file και γεμίζει το άρθρο με withespace characters. page\_remover.c δεν είναιthreaded όπως είναι το

bottleneck στο HDD IO speed.

2. sql-clear.sh

sql-clear.sh είναι ένα μικρό bash script που truncates όλους τους πίνακες από την βάση δεδομένων.

Με τον όρο Truncating εννοούμε ότι αφήνει τα table scheamata ανεπηρέαστα και διαγράφει όλη τα internalδεδομένα.

3. utf8thread.c

utf8thread.c είναι ένα άλλο χαμηλού επιπέδου πρόγραμμα το οποίο blanks out όλα τα invalid

utf-8 charactersαπό το φάκελο. Χρησιμοποιούμε pthreads για να επιταχύνουμε τα πράγματα.

4. webmonitor.py

webmonitor.py είναι ένα python script το οποίο sets up μια σελίδα web page που δείχνει

live δεδομένασε μορφή ιστογράμματος για την πρόοδο του πληθυσμού της βάσης δεδομένων.

. webmonitor.py serves a static html page and then deeds

it the data over websocket. Webmonitor μπορεί να δείχνει οποιοδήποτε stream από <epoc

date> <float value> pairs που λαμβάνει στοinput. Σαν παράδειγμα:

$ pip install tornado

Πρώτα πρέπει να εγκαταστήσουμε τα dependencies του script. Το οποίο μπορεί να είναι tornado, anasynchronous web framework supporting websockets. Δίνουμε οδηγίες

tornado, tornado θα υπηρετεί τις ακόλουθες σελίδες:

<! DOCTYPE HTML PUBLIC " -// W3C // DTD␣ HTML ␣ 4.01// EN"

" http :// www.w3.org/TR/ html4 / strict .dtd">

<html >

<head >

<meta http - equiv =" Content - Type " content =" text

/ html ;␣ charset =utf -8">

<title > DrNinjaBatmans Websockets </ title >

<script type =" text / javascript " src=" http ://

code . jquery . com/ jquery -1.10.1. js"></ script

>

<script type =" text / javascript " src=" http ://

code . highcharts . com/ highcharts .js"></

script >

<script >

var chart ; // global

var url = location . hostname + ’:’ + (

parseInt ( location . port ));

var ws = new WebSocket (’ws :// ’ + url + ’/

websocket ’);

ws. onmessage = function (msg) {

add\_point ( msg. data );

};

// ws. onclose = function () { alert (’

Connection closed .’); };

var add\_point = function ( point ) {

var series = chart . series [0] ,

shift = series . data . length > %d;

chart . series [0]. addPoint ( eval ( point ),

true , shift );

};

62

$( document ). ready ( function () {

chart = new Highcharts . Chart ( JSON . parse

(’%s ’));

});

</ script >

</ head >

<body >

<div id=" container " style =" width :␣800 px;␣

height :␣ 400 px;␣ margin :␣0␣ auto "></div >

</ body >

</ html >

Με την έννοια αυτή η σελίδα αναμένεται να διαβάζει ένα stream of values από ένα websocket

at ws://localhost:8888/hostname – αν και είναι αρκετά έξυπνο να αλλάξουμε το localhost:8888 εάν υπηρετεί αυτό μια άλλη τοποθεσία

– και να κάνουμε plot αυτά σε πραγματικό χρόνο χρησιμοποιώντας highcharts.js.

Ο πιθανός αναγνώστης ίσως παρατηρήσει ότι τα παραπάνω δεν είναι ακριβώς HTML αλλά περισσότερο ένα python formatted string. Αυτό συμβαίνει για 2 λόγους. Πρώτον γιατί το First script handles το configuration (βλέπε chart = new Highcharts.Chart(JSON.parse(’%s’Δεύτερον , το πλάτος του graph υπολογίζεται σε page loadtime και το plot χρειάζεται να μετατιπισθεί για να δείξει μόνο τα πιο τελευταία σημεία.

$ for i in {1..100}; do echo $i; sleep 1; done

| \

awk -oL "{ print ␣\$1 /100} " | \

python webmonitor .py

Αυτό θα παράγει σε διάστημα 1 δευτερολέπτου, αριθμούς από το 1 ως το100. Μετά τα normalizes χρησιμοποιώντας awk και τα τροφοδοτεί σε webmonitor. Αφού αυτή η εντολή εκτελεσθεί μπορούμε να ανοίξουμε τον browser και να κάνουμε navigate στο

localhost:8888.

Χρησιμοποιούμε αυτό για να ρυθμίσουμε από απόσταση το ολικό μέγεθος των δεδδομένωνπου το mysql καταναλώνει.

5. xml-parse.sh

Chapter 15

Mediawiki Extensions

Γιατη MediaWiki για να ενεργήσει όπως η wikipedia απαιτούνται μια σειρά από extensions. Η διαδικασία εγκατάστασης των εν λόγω extensions δεν είναι αυτοματοποιημένη ή streamline. Για να γίνει αυτόματη διαχείριση αυτής της πολυπλοκότητας ένας μηχανισμός παρέχεται την εγκατάσταση των extensions. Για υποστηρίξουμε επιπλέον για την wikipediabase πρέπει να προσθέσουμε τον ακόλουθο κώδικα στο Makefile.mwextnesions

(τροποποιημένο αναλόγως)

MW\_EXTENSIONS += newextension

mw - newextension -url = url/to/new/ extnesion / package .

tar.gz

mw - newextension -php = NewExtensionFile .php

mw - newextension - config = ’ $$phpConfigVariable = "

value ";’

Η wikipedia-mirror θα φροντίσει εάν τοextension είναι ήδη εγκαταστημένο και εάν δεν είναι θα τοποθετήσει τους σωστούς φακέλου στο σωτσό μέρος και θα διορθώσει τους κατάλληλους

configurationφακέλους. Τα entry points για την διαχείριση των extensions

είναι (με την προυπόθεση ότι τα ονόματα των εγγραφομένων extension είναι newextension):

make mw -print - registered - extensions *# Output a list*

*of the registed extensions*

make mw - newextension - enable *# Install and/or*

*enable the extension*

make mw - newextension - reinstall *# Reinstall an*

*extension*

make mw - newextension - disable *# Disable the*

*extension*

make mw - newextension - clean *# Remove the*

64

*extension*

Όλα τα registered extensions θα εγκατασταθούν και enabled όταν το wikipedia-

mirror έχει χτισθεί.

Chapter 16

Dumps

Η Wikipedia παρέχει μηνιαία dumps από όλη την βάση δεδομένων. Η μεγάλη πλειονότητα των

dumps είναι σε XML format και χρειάζεται να κωδικοποιηθούν σε MySQL γαι να φορτωθούν στη

βάση δεδομένων της wikipedia-mirror. Υπάρχουν πάνω από ένας τρόποι για να γίνει αυτό.There

1. PHP script

Η Mediawiki ships με ένα utility για να εισάγει τα XML dumps. Όμως η χρήση τους για εισαγωγή μιας συνολικής wikipedia mirror είανι αποθαρρυντική λόγω της εκτέλεσης τωνis

tradeoffs. Αντίθετα συνιστούνται άλλα εργαλεία όπως το mwdumperare τα οποία μετατρέπουν το XML dump σε MySQL queries που είναι και δημοφιλής βάση δεδομένων

2. mwdumper

Το συνιστώμενο εργαλείο που μεταφράζει τα XML dumps σε κώδικα MySQL code

είανιmwdumper. Mwdumper είναι γραμμένο σε java και είναι and shipped ξεχωριστά από τη mediawiki.

(a) Xml sanitizer

(b) Article dropper

Chapter 17

The xerces bug

Ότα γράφαμε τοmwdumper ένα περίεργο , semi-random bug εμφνίστηκε. While make

sql-dump-parts is running the following is encountered:

...

376 ,000 pages (14 ,460.426/ sec), 376 ,000 revs

(14 ,460.426/ sec)

377 ,000 pages (14 ,458.848/ sec), 377 ,000 revs

(14 ,458.848/ sec)

Exception in thread " main " java . lang .

ArrayIndexOutOfBoundsException : 2048

at org . apache . xerces . impl .io. UTF8Reader . read (άγνωστη πηγή

at org . apache . xerces . impl . XMLEntityScanner .

load (άγνωστη πηγή)

at org . apache . xerces . impl . XMLEntityScanner .

scanContent (άγνωστη πηγή)

at org . apache . xerces . impl .

XMLDocumentFragmentScannerImpl . scanContent

(άγνωστη πηγή)

at org . apache . xerces . impl .

XMLDocumentFragmentScannerImpl$FragmentContentDispatcher

. dispatch (άγνωστη πηγή)

at org . apache . xerces . impl .

XMLDocumentFragmentScannerImpl .

scanDocument (άγνωστη πηγή)

at org . apache . xerces . parsers .

XML11Configuration . parse (άγνωστη πηγή)

at org . apache . xerces . parsers .

XML11Configuration . parse (άγνωστη πηγή)

67

at org . apache . xerces . parsers . XMLParser . parse (άγνωστη πηγή)

at org . apache . xerces . parsers .

AbstractSAXParser . parse (άγνωστη πηγή)

at org . apache . xerces . jaxp .

SAXParserImpl$JAXPSAXParser . parse

at javax .xml. parsers . SAXParser . parse (άγνωστη πηγή)

SAXParser . java :392)

at javax .xml. parsers . SAXParser . parse (

SAXParser . java :195)

at org . mediawiki . importer . XmlDumpReader .

readDump ( XmlDumpReader . java :88)

at org . mediawiki . dumper . Dumper . main ( Dumper .

java :142)

make : \*\*\* [/ scratch / cperivol / wikipedia - mirror / drafts /

wikipedia - parts /enwiki -20131202 - pages - articles20 .

xml - p011125004p013324998 .sql] Error 1

Inspecting the makefules and running make --just-print sql-dump-parts

we find out that the failing command is:

$ java -jar / scratch / cperivol / wikipedia - mirror / tools /

mwdumper .jar -- format = sql :1.5 / scratch / cperivol /

wikipedia - mirror / drafts / wikipedia - parts /enwiki

-20131202 - pages - articles20 .xml -

p011125004p013324998 .fix.xml > / root / path /

wikipedia - parts // enwiki -20131202 - pages - articles20 .

xml - p011125004p013324998 .sql

Ευτυχώς αυτό δεν τρέχει για μεγάλο διάστημα έτσι ασφαλώς πειραματιζόμαστε.

Εδώ είναι τοtime output:

26.65 s user 1.73 s system 78% cpu 35.949 total

Το λάθος φαίνεται να είναι κατά την διάρκεια του διαβάσματος του XML dump έτσι δεν είναι ειδικό για το SQL output. Αυτό θα μπορούσε να είναι χρήσιμο για να διαπιστώσουμε πιο άρθρο προκαλεί το λάθος, αποσύροντας το ελπίζοντας να λυθεί το πρόβλημα

Για να το εντοπίσουμε κατ αρχάς προσπαθήσαμε να exporting στο XML:

$ java -jar / scratch / cperivol / wikipedia - mirror / tools /

mwdumper .jar -- format = xml / scratch / cperivol /

wikipedia - mirror / drafts / wikipedia - parts /enwiki

68

-20131202 - pages - articles20 .xml -

p011125004p013324998 .fix.xml > / tmp/just -a- copy .

xml

Όπως αναμενόταν όπως παραπάνω το ίδιο λάθος εμφανίστηκε. Στη συνέχεια κοιτάξαμε τα τελευταία δύο άρθρα προσπαθώντας να κάνουμε export τυπώνοντας με αντίστροφη σειρά το xml file,

βρίσκοντας τις τελευταίες δύο εμφανίσεις του *<title>* με grep και αναστρέφοντας ξανά να τυπώσουμε με την αρχική σειρά

(σημείωση ότι tac είναι όπως cat, μόνο πoυ εμφανίζει lines με αντίστροφη σειρά):

$ tac /tmp/just -a- copy .xml | grep "<title >" -m 2 |

tac

<title >The roaring 20s </ title >

<title > Cranopsis bocourti </ title > *# <- This is*

*the last one*

*Αυτή η λειτουργία τελειώνει γρήγορα παρά το ότι το /tmp/just-a-copy.xml είναι αρκετά*

*μεγάλο γιατί tac ψάχνει το τέλος του αρχείου και διαβάζει προς τα πίσω μέχρι το grep*

*να βρει τα 2 περιστατικά που ψάχνει για και κλείνει. Στο ext3 η αναζητούμενη εργασία*

*δεν διασχίζουν ολόκληρο το αρχείο . Πράγματι, από τον κώδικα tac:*

if ( lseek ( input\_fd , file\_pos , SEEK\_SET ) < 0)

error (0, errno , \_("%s:␣ seek ␣ failed "), quotef (

file ));

*/\* Shift the pending record data right to make room*

*for the new.*

*The source and destination regions probably*

*overlap . \*/*

memmove ( G\_buffer + read\_size , G\_buffer ,

saved\_record\_size );

past\_end = G\_buffer + read\_size + saved\_record\_size ;

*/\* For non - regexp searches , avoid unnecessary*

*scanning . \*/*

if ( sentinel\_length )

match\_start = G\_buffer + read\_size ;

else

match\_start = past\_end ;

if ( safe\_read ( input\_fd , G\_buffer , read\_size ) !=

read\_size )

{

error (0, errno , \_("%s:␣ read ␣ error "), quotef (

file ));

return false ;

}

Ασ σώσουμε την διαδρομή από το αρχικό xml αρχείο σε μια μεταβλητή γιατί το χρησιμοποιούμε πολύ.

Έτσι από δω και πέρα το $ORIGINAL\_XML θα έχει διαδρομή από το αρχικό xml.

$ export ORIGINAL\_XML =/ scratch / cperivol / wikipedia -

mirror / drafts / wikipedia - parts /enwiki -20131202 -

pages - articles20 .xml - p011125004p013324998 .fix. xml

Πρώτα ασ δούμε αν κάτι περίεργο συμβαίνει με το xmlαρχείο:

$ grep "<title > Cranopsis ␣ bocourti </ title >" -A 200 -B

100 $ORIGINAL\_XML | less

| less is to browse and -A 200 -B 100 means *"show 200 lines after and*

*100 before the matching line"*. Τίποτα περίεργο δεν βρέθηκε, έτσι δεν μπορούμε πραγματικά να διορθώσουμε το πρόβλημαin-place. Θα προσπαθήσουμε να αποσύρουμε βίαια ολόκληρο το άρθρο και ελπίζουμε ότι θα δουλέψει

(spoiler alert: it does).

Θα προσπαθήσουμε να επιθεωρήσουμε τους parents του τίτλου από το σπασμένο άρθρο..

Ευτυχώς το xml που δημιουργήθηκε είναιindented έτσι μπορούμε να βρούμε τους parents που βασίζονται σ αυτό. Αριθμήσαμε 6 spaces από indentation έστι θα ψάξουμε προς τα πίσω από εκεί σε κάθε επίπεδο του indentation. Η πρώτη γραμμή που βρήκαμε σε κάθε περίπτωση θα αποτελεί έναν άμεσο parentτου άρθρου.

$ for i in {0..6}; do \

echo " Level ␣$i:"; \

tac /tmp/just -a- copy .xml | grep "^␣\{ $i \} <[^/] " -

m 1 -n | tac ; \

done

Level 0:

17564960: < mediawiki xmlns =" http :// www. mediawiki .org /

xml/export -0.3/ " xmlns :xsi =" http :// www.w3. org

/2001/ XMLSchema - instance " xsi : schemaLocation =" http

:// www. mediawiki .org/xml/export -0.3/ ␣ http :// www.

mediawiki .org/xml/export -0.3. xsd " version =" 0.3"

xml: lang ="en">

Level 1:

Level 2:

38: <page >

Level 3:

70

Level 4:

35: <revision >

Level 5:

Level 6:

26: <text xml: space =" preserve " >&lt;!-- This

article was auto - generated by [[ User : Polbot ]]. --&

gt;

Φαίνεται ότι το xml είναι σελίδα σε ένα grand domain που ονομάζεται mediawiki. Θα μπορούσαμε να δούμε αυτό επίσης και από την java source too αλλά αν και πιο ακριβό είναι πιο γρήγορο να χρησιμοποιούμε το source του mwdumper.

Ο πιο εύκολος τρόπος να κόψουμε αυτό το άρθρο θα ήταν το awk αλλά θα πάρει πολύ χρόνο και εμείς θέλουμε να βελτιστοποιήσουμε και να αυτοματοποιήσουμε την όλη διαδικασία. Πρώτα ας προσπαθήσουμε απλώς να συγκρίνουμε τα άρθρα :

$ cmp /tmp/just -a- copy .xml $ORIGINAL\_XML

/tmp/just -a- copy .xml / scratch / cperivol / wikipedia -

mirror / drafts / wikipedia - parts /enwiki -20131202 -

pages - articles20 .xml - p011125004p013324998 .fix. xml

differ : byte 2, line 1

Αυτό ήταν γρήγορο. . . Ας δούμε πιο ήταν το λάθος:

$ head $ORIGINAL\_XML

<mediawiki xmlns =" http :// www. mediawiki .org/xml/export

-0.8/ " xmlns :xsi=" http :// www .w3.org /2001/ XMLSchema

- instance " xsi: schemaLocation =" http :// www .

mediawiki .org/xml/export -0.8/ ␣ http :// www . mediawiki

.org/ xml/export -0.8. xsd" version ="0.8 " xml: lang ="

en">

<siteinfo >

<sitename > Wikipedia </ sitename >

<base > http :// en. wikipedia . org/ wiki / Main\_Page </

base >

<generator > MediaWiki 1.23 wmf4 </ generator >

<case >first -letter </ case >

<namespaces >

<namespace key=" -2" case ="first - letter ">Media </

namespace >

<namespace key=" -1" case ="first - letter ">Special

</ namespace >

<namespace key="0" case ="first - letter " />

71

$ head / tmp/just -a- copy .xml

<?xml version =" 1.0" encoding ="utf -8" ?>

<mediawiki xmlns =" http :// www. mediawiki .org/xml/export

-0.3/ " xmlns :xsi=" http :// www .w3.org /2001/ XMLSchema

- instance " xsi: schemaLocation =" http :// www .

mediawiki .org/xml/export -0.3/ ␣ http :// www . mediawiki

.org/ xml/export -0.3. xsd" version ="0.3 " xml: lang ="

en">

<siteinfo >

<sitename > Wikipedia </ sitename >

<base > http :// en. wikipedia . org/ wiki / Main\_Page </

base >

<generator > MediaWiki 1.23 wmf4 </ generator >

<case >first -letter </ case >

<namespaces >

<namespace key=" -2">Media </ namespace >

Η συνυσφορά των xml tags είναι αρκετά διαφορετική. Το καλύτερό μας αποτέλεσμα είναι άν τα

line numbers συμπίπτουν. Μετρήσαμε τον αριθμό των γραμμών στο /tmp/just-a-copy.xml

και ελπίζουμε ότι ο αντίστοιχος αριθμός των γραμμών στο $ORIGINAL\_XML θα είναι ίδιος.

Εάν αυτό συμβεί μπορούμε να αγνοήσουμε τις contextual xml πληροφορίες και να σβήσουμε το προβληματικό άρθρο. Θα χρησιμοποιήσουμε wc το οποίο είναι αρκετά γρήγορο.

fast.

$ wc -l / tmp/just -a- copy .xml

17564961 / tmp/just -a- copy .xml

And the corresponding line in $ORIGINAL\_XML would be about:

$ sed " 17564960 q;d" $ORIGINAL\_XML

[[ Willie Jones ( American football )| Willie Jones ]],

Ποδόσφαιρο(football)....καμιά σχέση με τα βατράχια (frogs). Φαίνεται ότι δεν μπορούμε να αποφύγουμε κάποιο επίπεδο του parsing.

1. Parsing

Θα κάνουμε τις ακόλουθες υποθέσεις για να αποφύγουμε το κατάλληλο parsingτου εγγράφου:

• Το XML στο αρχικό αρχείο είναι i the original file is valid

• Κάθε XML μέσα στο άρθρο είναιHTML escaped

Κατ αρχάς δουλεύοντας με lines είανι αργή διαδικασία γιατί το user space code χρειάζεται να κοιτά για newlines. Δουλεύοντας με bytes αφιερώνει εργασία στο kernel, επιταχύνοντας την εργασία σημαντικά. Έτσι το dd είναι το σωστό εργαλείο για την συγκεκριμένη δουλειά.

Έτσι πρώτα θα βρούμε σε πιο byte είναι το άρθρο που μας ενδιαφέρει.

$ grep -b "<title > Cranopsis ␣ bocourti </ title >" -m

1 $ORIGINAL\_XML

1197420547: <title > Cranopsis bocourti </ title >

Αυτό ίσως πρει καποιο χρόνο αλλά δυστυχώς είναι η μόνη μας επιλογή.

Η στρατιγική μας είναι να φτιάξουμε 2 αρχεία: το /tmp/original\_tail.xml το οποίο να περιέχει όλα τα δεδομένα που υπάρχουν *μετά* τη σελίδα που θέλουμε να βγάλουμε και το /tmp/original\_head.xml το οποίο περιέχει όλα τα δεδομένα *πριν* τη σελίδα που θέλουμε να βγάλουμε.

Τώρα θα χρησιμοποιήσουμε sed να ψάξει για </page> after byte 1197420547 το οποίο

will be point *x* we will and dump the contents of $ORIGINAL\_XML after

point *x*:

$ dd if= $ORIGINAL\_XML skip =1197420547 ibs =1 | sed

’0 ,/ <\/ page >/d’ > / tmp/ original\_tail .xml

Θαυμάσια, αυτό δούλεψε! το dd δεν αντιγράφει αντίστροφα έτσι θα χρειασθούμε να κάνουμε κάτι πιο περίπλοκο για να κατασκευάσουμε /tmo/original\_head.xml. Ασ υποθέσουμε ότι η θέση που βρήκαμε τον τίτλο της σελίδας που θέλουμε να αφαιρέσουμε είναι *\_* = 1197420547 και το σημείο που η σελίδα αρχίζει είναι *\_*. Είναι ασφαλές να υποθέσουμε ότιI*\_ > \_􀀀*1000 (calibrate το constant 1000 εάν αυτή η υπόθεση είναι λάθος, αλλά τελικά δεν ήταν). Με αυτό τον τρόπο χρειάζεται μόνο να ψάξουμε στο 1Kb για <page>. Αποτελεσματικά αντί να κάνουμε

copy τα bytes στο εύρος [0*; \_*) εμείς concatenating δύο διακυμάνσεις

[0*; \_􀀀*1000]*[*(*\_􀀀*1000*; \_*) δημιουργώντας ένα subshell το οποίο θα έχει πρώτο output

το πρώτο εύρος και στη συνέχεια output (*\_ 􀀀* 1000*; \_*) stopping όταν βρεί <page>. Παρακάτω είναι ένα liner:

$ (dd count =$ ((1197420547 -1000) ) ibs =1 if=

$ORIGINAL\_XML ; \

dd if= $ORIGINAL\_XML count =1000 skip =$

((1197420547 -1000) ) ibs =1 \

| tac | sed ’/<page >/,$d ’ | tac) > /tmp/

original\_head .xml

73

2. Η τελική λύση

Όλα τα παραπάνω χρησιμοποιήθηκαν για να συντεθεί ένα script το οποίο lives στο data/xml-parse.sh

το οποίο χρησιμοποιήθηκε από το makefiles να απομακρύνει όλα τα προβληματικά άρθρα. Εάν το

mwdumper αποτύχει, ταυτοποιούμε το άρθρο που προκάλεσε το πρόβλημα και το απομακρύνουμε χρησιμοποιώντας xml-parse.sh. Στη συνέχεια rerun το mwdumper. Το επαναλαμβάνουμε αυτό We μέχρι το mwdumper πετύχει. Συνολικά τα προβληματικά άρθρα είναι περίπου10-15, και είναι διαφορετικά ανάλογα με το dump που χρησιμοποιείται.

3. Covering up with spaces

Από την παραπάνω έκθεση των τρόπωνε που αντιμετωπίσουμε το θέμα του

breaking article παραλείψαμε κάτι προφανές. Μια θεματικά διαφορετική προσέγγιση: covering up το breaking article με κενά. Μόλις εντοπίσουμε το εύρος στο οποίο η σελίδα βρίσκεται μπορούμε να κάνουμε mmap επακριβώς στο τμήμα του

$ORIGINAL\_XML και στη συνέχεια να κάνουμε memset covering it up με χαρακτήρες κεών.

Η εφαρμογή lives στο data/page\_remover.c,

Παρακάτω παρουσιάζουμε το call στο mmap:

ctx ->off = off - pa\_off ;

ctx ->fd = open (fname , O\_RDWR , 0 x0666 );

if (ctx ->fd == -1) {

perror (" open ");

return NULL ;

}

ctx -> size = len ;

ctx -> data = mmap (0, len+ctx ->off , PROT\_READ |

PROT\_WRITE ,

MAP\_SHARED , ctx ->fd , pa\_off );

if (ctx -> data == MAP\_FAILED ) {

perror (" mmap ");

return NULL ;

}

and the mmemset:

*/\* You MIGHT want to thread this but I dont think*

*it will make*

*\* much more difference than memset . \*/*

memset (ctx -> data + ctx ->off , ’␣’, ctx -> size );

Περιέργως αυτό δεν έλυσε το πρόβλημα του mwdumper issue το οποίο δείχνει ότι μάλλον προκειται για πιθανό memory leak στο τμήμα του xerces αλλά αυτό ξεπερνά τους στόχους της παρούσας εργασίας να κάνει διόρθωση debug εάν έχουμε την επιλογή.

4. The sed command

Παραπάνω αναφέραμε την χρήση της εντολής sed και ίσως ελιναι χρήσιμο να το αναπτ΄θξουμε περεταίρω.

. Sed είναι ένα unix εργαλείοπου βρίσκεται στο coreutils το οποίο σύμφωνα με το man page είναι ένας stream editor που φιλτραρει και μεταμορφώνει text.

η βασική λειτουργία είναι ότι το *"pattern space"*, ή τοinput stream

το οποίο είναι ένα unix stream που έρχεται από το αρχείο, ένα pipe ή μόνο stdin, περνά μέσα από μια programmable pipeline. Εκτυπώνεται ή αυτούσιο το modified pattern

space ή, με τη χρήση του -n flag, επιλεγμένα τμήματα αυτού.

Ας δούμε τη χρήση που κάναμε παραπάνω για το sed

Αρχικά χρησιμοποιήσαμε sed για να εκτυπώσουμε μια μεμονωμένη line σε ένα αρχείο:

$ sed " 17564960 q;d"

Αυτό τοsed προγραμμα διαχωρίζεται από semicolon. Sed iterates τις

lines του input stream και τρέχει καθένα από τις; διαχωρισμένες εντολές σε σειρά μέχρι μια να επιτύχει. Οι εντολές εδώ είναι

17564960q και d. 17564960q θα σταματήσει το sed όταν φτάσει στη line 17564960. d θα απορρίπτει την παρούσα. Έστι το sed απορρίπτει lines μέχρι να συναντήσει την line 17564960 την οποία εκτυπώνει και τελειώνει.

Χρησιμοποιούμε μια εντολή sed ως μέρος μιας σειράς εντολών shell

piped όλες μαζί με στόχο να εκτυπωθούν όλες ο lines ενός stream μετά από ένα συγκεκριμένο μοτίβο

(στην περίπτωσή μας </page>).

$ sed ’0 ,/ <\/ page >/d’

αυτή τη φορά είχαμε μόνο μια εντολή sed, d. Sed iterates στις

lines στο stream, απορρίπτοντας lines στο εύρος των lines 0 στηline

που ταυτίζεται με το <\/page>, αποτελεσματικά τυπώνει μόνο lines μετά το </page>.

Η τελική μας χρήση του sed iείναι η the inverse of the aforementioned one,

$ sed ’/<page >/,$d ’

Εδώ το sed iterates ξανά σε όλες τις lines τουstream. Αυτή τη φορά απορρίπτοντας lines ανάμεσα στην πρώτη line that matches <page>μέχρι την τελικήline, σημειωμένη με $.

Chapter 18

Automation

Δημιουργώντας μια wikipedia mirror ίσως φαίνεται μια απλή διαδικασία αλλά συμπεριλαμβάνει πολλούς caveats, nuances και επαναλαμβανόμενα tasks. Πολλαπλοί μέθοδοι αυτοματισμού εφαρμόσθηκαν για να ολοκληρώσουν μια μεγάλη ποικιλία tasks που συμπεριλαμβάνονται στην εκτέλεση.

1. Makefiles / laziness

Το πιο σημαντικό μέρος του αυτοματισμού της wikipedia-mirror είναι το make

build system. Make είναι ένα build system όπου κάποιος μπορεί να δηλώσει τα απαιτούμενα αρχεία (targets), dependencies για αυτά, και ένα set από shell commands που θα χτίσουν

αυτά ταtargets. Κάθε target είναι ουσιαστικά μια finite state machine

με δύο states:

• Ένα αρχείο που υπάρχει και είναι επικυροποιημένο με τα dependencies και

• Ένα αρχείο που είτε δεν υπάρχει ή η modification date είναι παλαιότερηαπό αυτό ή τουλάχιστον ενός από τα dependencies.

Και μια σειρά απόshell εντολές για την μεταφορά από την παλαιότερη στην παρούσα κατάσταση.to

Για παράδειγμα , σώζουμε το ακόλουθο ως Makefile σε μια εργασία που περιέχει τα αρχεία

foo.c, foo.h, bar.c and bar.h:

foo.o: foo.c foo .h

gcc foo.c -c -o foo .o

bar.o: bar.c

gcc bar.c -c -o bar.o

foobar : foo .o bar.o

gcc foo.o bar.o -o foobar

αυτό σημαίνει ότι για να χτίσουμε foobar χρειαζόμαστε foo.o και bar.o. Και για να χτίσουμε foo.o και bar.o χρειαζόμαστε foo.c και foo.h, και bar.c και bar.h αντίστοιχα.

Επίσης παρέχουμε εντολές για να χτισθεί το foo.o, bar.o και

foobar, οι οποίες είναι

• gcc foo.c -c -o foo.o

• gcc bar.c -c -o bar.o

• and gcc foo.o bar.o -o foobar

αντίστοιχα. παρατηρούμε ότι δεν υπάρχουν κανόνες για τα .c και .h αρχεία.

TΑυτό συμβαίνει γιατί το make πρέπει να αποτυγχάνει εάν δεν είναι παρόντα. Έτσι εάν τρέχουμε το make foobar, το make θα ελέγχει για την ύπαρξη του foobar και την ημερομηνία της τροποποίησης.

Εάν το foobar λείπει ή η ημερομηνία τροποποίησης είναι προηγούμενη από τα

deopendecie’ του (πχ foo.o και bar.o) αυτό θα ξαναχτιστεί. Εάν κα΄ποιο από τα If any dependecies απουσιάζει η ίδια λογική ισχύει και για αυτό.

Με αυτό τον τρόπο εάν χτίσουμε μια φορά το

foobar , και μετά κάνουμε edit bar.c και rerun make foobar, το make θα

recursively αφαιρέσει αυτό

• bar.o είναι out of date όσον αφορά την dependency bar.c

• Όταν bar.o έχει πλέον μια πιο πρόσφατη ημερομηνία μετατροπής από το foobar και για αυτό το τελευταίο είναι out of date όσον αφορά την dependencyτου bar.o, έτσι χρειάζεται να ξανχτιστεί.

Με αυτό τον τρόπο make πετυχαίνει μια σχεδόν βέλτιστη στρατηγική για την επίτευξη κάθε φορά του ελάχιστου ποσοστού των απαιτούμενων στόχων . Τώρα που ξεκαθαρίσαμε την βασική λογική των make ας κάνουμε πιο σαφή μερικά από τα βασικά χαρακτηριστικά τους που κάνουν τη ζωή μας πιο εύκολη.

(a) Phony targets

Μερικές εργασίες δεν είναι αρχεία και χρειάζονται να τρέχουν κάθε φορά που το make τις συμπεριλαμβάνει στο dependency tree. Γι αυτά έχουμε ένα ειδικό keywork .PHONY:.

Εδώ είναι ένα παράδειγμα.

. PHONY :

clean :

rm -rf \*

78

Αυτό λέι στο make ότι κανένα αρχείο ονομαζόμενο clean will emerge from running

rm -rf \*, και επίσης ακόμα και εαν υπάρχει ένα up-to-date ονομαζόμενο cleanαρχείο, αυτό τοtarget θα τρέχει ανεξάρτητα.

Αξίζει να σημειώσουμε ότι τα phony dependencies πάντα θα render το

dependent target out-of-date.

Για παράδειγμα:

. PHONY :

say - hello :

echo " hello "

test .txt: say - hello

touch test .txt

Όταν το touch test.txt θα τρέχει κάθε φορά που τρέχουμε make test.txt

απλώς γιατί το make δεν μπορεί να γνωρίζει με βεβαιότητα ότι το phony target say-hello

δεν άλλαξε τίποτε σημαντικό για το test.txt. Για αυτό το λόγο τα

phony targets χρησιμοποιούνται για user facing tasks.

(b) Variables

Τα makefiles cμπορούν να έχουν μεταβλητές ορισμένες με ποικίλους τρόπους.

Μερικές καταστάσεις που έχουν γίνει για να χρησιμοποιηθούν στην wikiepedia-mirror παρουσιάζονται παρακάτω.

i. Simple variables

Μερικές φορές δεν είναι επιθυμητό για τις μεταβλητές να είναι expanded indefinitely:

kurma = the world $( support1 )

animal1 = four elephants

animal2 = tortoise

support1 = supported by $( animal1 ) $(

support2 )

support2 = supported by a $( animal2 ) $(

support2 )

all:

echo $( kurma )

So what would makefile have said to Bertrand Russell [re-

frence] (or whoever)?

$ make --just - print

79

Makefile :5: \*\*\* Recursive variable ‘

support2 ’ references itself ( eventually

). Stop .

Έτσι κατά κάποιον τρόπο το variable system των makeείναι συνολικό [reference].

As the hindus "solved" this problem so can we in makefiles:

kurma = the world $( support1 )

animal1 = four elephants

animal2 = tortoise

support1 = supported by $( animal1 ) $(

support2 )

support2 := supported by a $( animal2 ) $(

support2 )

all:

echo $( kurma )

Και όταν τρέχουμε παίρνουμε:

make --just - print

echo the world supported by four elephants

supported by a tortoise

So basically support2 is removed from scope when the support2

variable is substituted.

ii. Automatic variables

Το Makefile επίσης ορίζει μερικές contextual variables οι οποίες είναι ορισμένες. Οι automatic variables ορισμένες από το gnu make είναι οι ακόλουθες:

• $@: Το όνομα του αρχείου τουtarget of the rule. Εάν το target

είναι ένα archive member, τότε $@ είναι το όνομα του archive

αρχείου. Στο pattern rule που έχει πολλαπλά targets (βλέπε Intro-

duction στο Pattern Rules), $@ είναι το όνομα του οποιουδήποτε

target που κάνει το rule’s recipe να τρέχει.

• $%: Το όνομα τουtarget member, όταν το target είναι ένα

archive member. (Βλέπε Archives). Για παράδειγμα, εάν το tar-

get είναι foo.a(bar.o) τότε %% είναι bar.o και $@ είναι foo.a. $% είναι

άδειο όταν το target δεν είναι ένα archive member.

• $<: Το όνομα του πρώτουprerequisite. Εάν το target πήρε

το recipe του από έναν implicit rule, αυτό θα είναι το πρώτο prerequisite που προστέθηκε από το implicit rule (βλέπε Implicit Rules).

• $?: Τα ονόματα από όλα τα prerequisites που είναι νεότερα από το target, με κενά μεταξύ τους . Για τα prerequisites που είναι archive members, μόνο named member χρησιμοποιούνται(βλέπε Archives).

• $^: Τα ονόματα όλων των prerequisites, με κενά μεταξύ τους. Για τα prerequisites τα οποία είναι archive members, μόνο των named member χρησιμοποιείται (βλέπε Archives). ένα

target έχει μόνο ένα prerequisite σε κάθε άλλο αρχείο από το οποίο εξαρτάται, αναξαρτήτως από το πόσες φορές κάθε αρχείο είναι καταχωρημένο ως ένα no matter how many times each file prerequisite. Έτσι εάν τοποθετήσουμε στη λίστα ένα prerequisite για περισσότερο από μια φορά για ένα target, η value του $ˆ περιέχει μόνο ένα αντίγραφο του ονόματος.

Αυτή η λίστα δεν περιέχει κανένα από το order μόνο

prerequisites; Για εκείνα που βλέπουν την $| variable, πιο κάτω.

• $+: Αυτό είναι όπως το $^, αλλά οι prerequisites που είναι στη λίστα περισσότερες από μία φορές είναι duplicated με τη σειρά που είναι στη λίστα του makefile. Αυτό αρχικά είναι χρήσιμο για την χρήση στο να linking commands όπου έχει νόημα να επαναλαμβάνονται ταlibrary file names με μια καθορισμένη σειρά.

• $| Τα ονόματα όλων των order-only prerequisites, με κενά μεταξύ τους.

• $\*: Το stem με το οποίο ένα implicit rule matches (βλέπε

How Patterns Match). Εάν το target είναι dir/a.foo.b και το

target pattern is a.%.b τότε το stem είναι dir/foo. Το

stem είναι χρήσιμο για να δημιουργηθούν ονόματα των σχετικών αρχείων. Στο

static pattern rule, το stem είναι μέρος του ονόματος του αρχείου το οποίο ταιριάζει το

% στο target pattern. Στο explicit rule,

δεν υπάρχει stem; έτσι $\* δεν μπορεί να προσδιορισθεί με αυτόν τον τρόπο.

Αντίθετα , εάν το όνομα τουtarget name τελειώνει με ένα αναγνωρισμένο suffix

(βλέπε Old-Fashioned Suffix Rules), $\* είναι το όνομα τουtarget

μείον τοsuffix. Για παράδειγμα, εάν το όνομα του target είναι

foo.c, τότε $\* είναι τοποθετημένο στο foo, όταν .c είναι ένα suffix. Το GNU

make αυτό το περίεργο πράγμα μόνο για compatibility με άλλες

implementations του make. Πρέπει γενικά να αποφεύγουμε την του $\* εκτός από τα implicit rules ή τα static patternrules. Εάν τοIόνομα του target σε ένα explicit rule δεν τελειώνει με ένα αναγνωρίσιμο

suffix, $\* είναι τοποθετημένο στο empty string για αυτό το rule.

81

(c) Functions

2. Bitnami

Chapter 19

Performance

1. Compile time

Το Compile time περιλαμβάνει το χρόνο που χρειάζεται για:

• Downloading όλα τα στοιχεία του wikipedia server

• The bitnami stack

– mwdumper

– mediawiki-extensions

– Εγκαθιστά και χτίζει αυτά τα στοιχεία (~1 min)

– Downloading the wikipedia dumps

– Preprocessing the dumps (~10 mins)

– Populating τη mysql βάση δεδομένων(~10 days)

Τα Builds έγιναν στο d Infolab’s Ashmore. Τα system’s specs

είναι quite high end αλλά το bottleneck ήταν το disk IO έτσι λιγότερο από 1% από τις υπόλοιπες διαθέσιμες πηγές χρησιμοποιήθηκαν κατά την διάρκεια του MySQL database population.

(a) Attempts to optimizing MySQL

2. Runtime

Το Runtime της wikipedia mirror φαίνεται να είναι αρκετά αργό ώστε να είναι χρήσιμο και γι αυτό η εργασία αυτή τελικά εγκαταλήφθηκε. Για ολόκληρο το wikipedia dump του Ιουλίου 2014 the load time γαι τον Barack Obama, χωρίς το πλεονέκτημα του caching ήταν περίπου ~30s.

83

Chapter 20

Appendix (script sources)

1. pageremover.c

*/\**

*\* Copyright 2014 Chris Perivolaropoulos <*

*cperivol@csail .mit .edu >*

*\**

*\* This program is free software : you can*

*redistribute it and/or*

*\* modify it under the terms of the GNU General*

*Public License as*

*\* published by the Free Software Foundation ,*

*either version 3 of the*

*\* License , or (at your option ) any later version*

*.*

*\**

*\* This program is distributed in the hope that*

*it will be useful , but*

*\* WITHOUT ANY WARRANTY ; without even the implied*

*warranty of*

*\* MERCHANTABILITY or FITNESS FOR A PARTICULAR*

*PURPOSE .*

*\**

*\* See the GNU General Public License for more*

*details . You should*

*\* have received a copy of the GNU General Public*

*License along with*

*\* this program .*

*\**

*\* If not , see <http :// www .gnu .org / licenses />.*

*\**

84

*\* This should fill a range in a file with spaces*

*. This is an in - place*

*\* operation so it should be pretty fast .*

*\**

*\* Usage : page\_remover PATH OFFSET LENGHT*

*\*/*

# include <assert .h>

# include <fcntl .h>

# include <pthread .h>

# include <stdio .h>

# include <stdlib .h>

# include <string .h>

# include <sys/ mman .h>

# include <sys/ stat .h>

# include <sys/ types .h>

# include <semaphore .h>

# include <unistd .h>

# include <unistd .h>

# define USAGE\_INFO " page\_remover ␣ PATH ␣ OFFSET ␣

LENGTH "

# define PRINT (ctx , args ...) do { sem\_wait (& ctx ->

stdio\_mutex ); \

printf ( args );

\

fflush ( stdout );

\

sem\_post (& ctx ->

stdio\_mutex );

\

} while (0)

typedef struct context {

int fd;

size\_t size ;

off\_t off;

sem\_t stdio\_mutex ;

void \* data ;

} context\_t ;

context\_t \* context\_init ( char \* fname , off\_t off ,

size\_t len)

{

85

context\_t \* ctx = ( context\_t \*) malloc ( sizeof (

context\_t ));

off\_t pa\_off = off & ~( sysconf ( \_SC\_PAGE\_SIZE )

- 1);

sem\_init (& ctx -> stdio\_mutex , 0 */\* Shared .*

*Usually ignored \*/* , 1);

PRINT (ctx , " Opening ␣%s␣at␣%lu␣( len:␣%lu)\n",

fname , off , len);

ctx ->off = off - pa\_off ;

ctx ->fd = open (fname , O\_RDWR , 0 x0666 );

if (ctx ->fd == -1) {

perror (" open ");

return NULL ;

}

ctx -> size = len ;

ctx -> data = mmap (0, len+ctx ->off , PROT\_READ |

PROT\_WRITE ,

MAP\_SHARED , ctx ->fd , pa\_off );

if (ctx -> data == MAP\_FAILED ) {

perror (" mmap ");

return NULL ;

}

return ctx ;

}

void context\_destroy ( context\_t \* ctx)

{

if ( close (ctx ->fd) == -1)

perror (" close ");

if ( munmap (( void \*) ctx ->data , ctx -> size ) ==

-1)

perror (" munmap ");

sem\_destroy (& ctx -> stdio\_mutex );

free (ctx);

}

int main (int argc , char \* argv [])

{

86

if ( argc != 4)

fprintf (stderr , USAGE\_INFO );

context\_t \*ctx = context\_init ( argv [1] , atoi (

argv [2]) , atoi ( argv [3]) );

*/\* You MIGHT want to thread this but I dont*

*think it will make*

*\* much more difference than memset . \*/*

memset (ctx -> data + ctx ->off , ’␣’, ctx -> size );

context\_destroy (ctx );

return 0;

}

2. utf8thread.c

# include <assert .h>

# include <fcntl .h>

# include <pthread .h>

# include <stdio .h>

# include <stdlib .h>

# include <string .h>

# include <sys/ mman .h>

# include <sys/ stat .h>

# include <sys/ types .h>

# include <semaphore .h>

# include <unistd .h>

# include <unistd .h>

sem\_t stdio\_mutex ;

# define PRINT ( args ...) do { sem\_wait (& stdio\_mutex )

; \

printf ( args );

\

fflush ( stdout );

\

sem\_post (& stdio\_mutex );

\

} while (0)

*/\* # define DEBUG ( args ...) PRINT (*

*args ) \*/*

87

# define DEBUG (...)

# define DEFAULT\_CHAR ’␣’

# define WORKERS 8

# define MESSAGE\_DENSITY 1000000000

typedef unsigned long long u64;

# define UTF\_LC (l) ((0 xff >> (8 - (l))) << (8 - (l

)))

# define UTF\_CHECK (l, c) ((( UTF\_LC (l) & (c)) ==

UTF\_LC (l)) && (0 == ((c) & (1 << (7 -(l))))))

# define UTF\_LEN (x) ( UTF\_CHECK (6, x) ? 6 : \

UTF\_CHECK (5, x) ? 5 : \

UTF\_CHECK (4, x) ? 4 : \

UTF\_CHECK (3, x) ? 3 : \

UTF\_CHECK (2, x) ? 2 : -1)

struct crange {

u64 start , end ;

};

*/\* Get return the next character after the last*

*correct one. \*/*

inline u64 valid\_utf8 ( u64 c)

{

char i;

*/\* Ascii \*/*

if ((\*( char \*)c & 0 x80) == 0)

return c+1;

*/\* \*/*

for (i = UTF\_LEN (\*( char \*)c) -1; i >0; i --) {

c++;

if (! UTF\_CHECK (1, \*( char \*)c)) {

return (u64 ) NULL ;

}

}

return i <0 ? 0 : c+1;

}

88

void \* fix\_range ( void \* \_r)

{

struct crange \* r = \_r;

u64 tmp , id = r-> start ;

long long unsigned count = 0;

while (( u64 )r-> start < (u64)r->end) {

if ( count ++ % MESSAGE\_DENSITY == 0)

printf ("[ worker :␣0x %016 llx]␣ Done ␣ with ␣

% lluK .\n", id , count % 1024) ;

if (!( tmp = valid\_utf8 (r-> start ))){

PRINT (" Invalid ␣ char ␣0x%x␣( next :␣0x%x)\n

",

\*( char \*)r->start , \*( char \*)(r->

start +1));

\*(( char \*)r-> start ) = DEFAULT\_CHAR ;

(r-> start )++;

} else {

r-> start = tmp ;

}

}

PRINT ("[ worker :␣0x %016 llx]␣ OUT\n", id);

return NULL ;

}

void run(u64 p, u64 sz)

{

int n, i;

u64 wsize ;

pthread\_t workers [ WORKERS ];

struct crange rngs [ WORKERS ];

wsize = sz/ WORKERS + 1;

printf (" Base ␣ address :␣0x %016 llx ,␣ step ␣ size :␣0

x %016 llx \n", p, wsize );

for (i=0; i< WORKERS ; i ++){

rngs [i]. start = p + wsize \*i;

rngs [i]. end = p + wsize \*i + wsize ;

PRINT (" Spawning ␣ worker ␣%d␣on␣ range ␣[0x %016

llx ,␣0x %016 llx),␣% llu␣ bytes ...", i, rngs

[i]. start , rngs [i]. end , wsize );

89

if ((n = pthread\_create ( workers +i, NULL ,

fix\_range , ( void \*)( rngs +i)))) {

PRINT (" FAIL \n");

perror (" worker ");

return ;

}

PRINT ("OK\n");

}

PRINT (" Wrapping ␣up ...\ n");

for (i=0; i< WORKERS ; i ++) {

PRINT (" Joining ␣ worker ␣%d ...", i);

pthread\_join ( workers [i], NULL );

PRINT ("OK\n");

PRINT (" Worker ␣%d␣ went ␣ through ␣% llu␣ bytes .\n

",

i, (u64) rngs [i]. end - ( u64) rngs [i].

start );

}

}

int main (int argc , char \* argv [])

{

int fd;

long long int sz , p;

struct stat buf;

sem\_init (& stdio\_mutex , 0 */\* Shared . Usually*

*ignored \*/* , 1);

fd = open ( argv [1] , O\_RDWR , 0 x0666 );

if (fd == -1) {

perror (" open ");

return 1;

}

fstat (fd , &buf );

sz = buf . st\_size ;

printf (" File ␣ size :␣0x %016 llx\n", sz);

p = (u64) mmap (0, buf. st\_size , PROT\_READ |

PROT\_WRITE , MAP\_SHARED , fd , 0);

if (p == -1) {

perror (" mmap ");

90

return 1;

}

run(p, buf. st\_size );

if ( close (fd) == -1) {

perror (" close ");

return 1;

}

if ( munmap (( void \*)p, buf. st\_size ) == -1) {

perror (" munmap ");

return 1;

}

sem\_destroy (& stdio\_mutex );

return 0;

}

3. sql-clear.sh

*#!/ bin/ bash*

MUSER ="$1"

MPASS ="$2"

MDB="$3"

MYSQL =$4

*# Detect paths*

AWK=$( which awk )

GREP =$( which grep )

if [ $*# -ne 4 ]*

then

echo " Usage :␣$0␣{MySQL -User - Name }␣{MySQL -

User - Password }␣{MySQL - Database - Name }␣{

MySQL ␣ executable ␣to␣use}"

echo " Drops ␣all ␣ tables ␣ from ␣a␣ MySQL "

exit 1

fi

TABLES =$( $MYSQL -u $MUSER -p$MPASS $MDB -e ’show

tables ’ | $AWK ’{ print $1}’ | $GREP -v ’^

91

Tables ’ )

for t in $TABLES

do

echo " Clearing ␣$t␣ table ␣ from ␣ $MDB ␣ database

... "

$MYSQL -u $MUSER -p$MPASS $MDB -e " truncate

␣ table ␣$t"

done

4. webmonitor.py

"""

Just ␣ feed ␣ pairs ␣of

<epoc ␣date >␣<float ␣value >

or␣ even ␣ just

<float ␣value >

One␣way␣to␣do␣ that ␣ would ␣be

␣␣␣␣$␣<cmd >␣ stdbuf ␣-oL␣awk ␣"{ print ␣\$1/ $$max }"␣|␣

python ␣ webmonitor .py

and␣I␣ will ␣ plot ␣ them ␣on␣ port ␣ 8888. ␣ This ␣ will ␣ also

␣ pipe ␣the ␣ input ␣ right

out␣to␣the␣ output .␣ Strange ␣ input ␣ will ␣be␣ ignored ␣

and ␣ piped ␣ this ␣way ,

but␣ this ␣ needs ␣to␣be␣ done ␣by␣awk␣ aswell ␣in␣the␣

above ␣ example .

"""

import sys

import json

import time

from threading import Thread

from collections import deque

import tornado . websocket as websocket

import tornado . ioloop

import tornado .web

92

HTML = """

<! DOCTYPE ␣ HTML ␣ PUBLIC ␣" -// W3C // DTD␣ HTML ␣ 4.01// EN"

␣" http :// www.w3. org/TR/ html4 / strict .dtd">

<html >

␣␣<head >

␣␣␣␣<meta ␣http - equiv =" Content - Type "␣ content =" text

/ html ;␣ charset =utf -8" >

␣␣␣␣<title > DrNinjaBatmans ␣ Websockets </ title >

␣␣␣␣<script ␣ type =" text / javascript "␣src =" http ://

code . jquery . com/ jquery -1.10.1. js"></ script >

␣␣␣␣<script ␣ type =" text / javascript "␣src =" http ://

code . highcharts . com/ highcharts .js"></ script >

␣␣␣␣<script >

var␣ chart ;␣//␣ global

var␣url␣=␣ location . hostname ␣+␣ ’:’␣+␣( parseInt (

location . port ));

var␣ws␣=␣new␣ WebSocket (’ws :// ’␣+␣ url␣+␣ ’/

websocket ’);

ws. onmessage ␣=␣ function (msg )␣{

␣␣␣␣ add\_point (msg. data );

};

//␣ws. onclose ␣=␣ function ()␣{␣ alert (’ Connection ␣

closed .’);␣};

var␣ add\_point ␣=␣ function ( point )␣{

␣␣␣␣var␣ series ␣=␣ chart . series [0] ,

␣␣␣␣␣␣ shift ␣=␣ series . data . length ␣>␣%d;

␣␣␣␣ chart . series [0]. addPoint ( eval ( point ),␣true ,␣

shift );

};

$( document ). ready ( function ()␣{

␣␣␣␣ chart ␣=␣new␣ Highcharts . Chart ( JSON . parse (’%s ’)

);

});

␣␣␣␣ </script >

␣␣ </head ><body ><div␣id =" container "␣ style =" width :␣

800 px;␣ height :␣ 400 px;␣ margin :␣0␣ auto "></div ></

body ></ html >

"""

93

config = {

’ visible\_points ’: 10,

’ py\_chart\_opts ’: { ’chart ’: { ’renderTo ’: ’

container ’,

’

defaultSeriesType

’: ’spline ’

},

’title ’: { ’text ’: ’

DrNinjaBatmans ␣ data ’},

’xAxis ’: { ’type ’: ’

datetime ’,

’

tickPixelInterval

’: ’150 ’},

’yAxis ’: { ’minPadding ’:

0.2 ,

’maxPadding ’:

0.2 ,

’title ’: {’text

’: ’Value ’,

’

margin

’:

80}

},

’series ’: [{ ’name ’: ’Data

’,

’data ’: []}]}

}

def date\_float (s):

try:

date , val = s. split ()

except ValueError :

val = s. strip ()

date = time . time ()

return int ( date ), float ( val)

94

def send\_stdin (fn= date\_float ):

for raw in sys. stdin :

sys. stdout . write (raw )

*# Ignore strange input .*

try:

jsn = json . dumps (fn( raw))

buf. append (jsn)

for w in websockets :

try:

w. write\_message (jsn)

except websocket .

WebSocketClosedError :

pass

except :

pass

for ws in websockets :

ws. close ()

class StdinSocket ( websocket . WebSocketHandler ):

def open ( self ):

for i in buf:

self . write\_message (i)

websockets . append ( self )

def closs ( self ):

websockets . remove ( self )

class MainHandler ( tornado . web. RequestHandler ):

def get( self ):

self . write ( HTML % ( int( config [’

visible\_points ’]) ,

json . dumps ( config [’

py\_chart\_opts ’])))

if \_\_name\_\_ == " \_\_main\_\_ ":

application = tornado . web. Application ([

(r"/", MainHandler ),

95

(r’/ websocket ’, StdinSocket ),

])

buf = deque ( maxlen = int( config [’ visible\_points

’]))

websockets = []

config [’args ’] = []

for a in sys. argv [1:]:

if ’=’ in a:

k, v = a. split (’=’, 1)

config [k] = v

else :

config [’args ’]. append (a)

Thread ( target = send\_stdin ). start ()

application . listen (8888)

tornado . ioloop . IOLoop . instance (). start ()

5. xml-parse.sh

*#!/ bin/ bash*

*#*

*# Simply removing specific articles fixes the*

*xerces error with*

*# UTF8 . If the articles are alone the error goes*

*away*

*# aswell . Extremely weird but that ’s life .*

*Fortunately the article is*

*# just a stub about some toad ( Cranopsis bocourti*

*)*

*#*

*# xml - parse .sh ORIGINAL\_XML*

*TITLE\_OF\_ARTICLE\_TO\_REMOVE [ inplace ]*

*#*

*# if ‘inplace ‘ is there the c program will be*

*used to cover the article*

*# with spaces . This is much faster . Should be*

*anyway . Otherwise the*

*# page is just ommited and the result is dumped*

*in stdout . Helping*

*# messages are dumped in stderr After this you*

*can run:*

*#*

96

*# java -jar tools / mwdumper . jar RESULTING\_XML --*

*format = sql :1.5 > SQL\_DUMP*

set -e

set -o pipefail

if [[ $*# -lt 2 ]]; then*

echo "xml - parse .sh␣ ORIGINAL\_XML ␣

TITLE\_OF\_ARTICLE\_TO\_REMOVE ␣[ inplace ]" 1 >&2

exit 0

fi

function my\_dd {

coreutils\_version =$(dd -- version | head -1 |

cut -d\ -f3 | colrm 2 2 )

if [[ $coreutils\_version -ge 822 ]]; then

eval "dd␣ iflag = count\_bytes ␣ iflag = direct ␣

oflag = seek\_bytes ␣ibs =1M␣$@"

else

echo " Your ␣ coreutils ␣may␣be␣a␣ bit␣ old␣(

$coreutils\_version ).␣822␣is␣the␣one␣ cool

␣ kids ␣use ." >&2

eval "dd␣$@␣ibs =1"

fi

}

ORIGINAL\_XML =$1

*# Dump a part of the file in sdout using dd.*

*# Usage :*

*# file\_range <filename > <first\_byte > <start |end|*

*length >*

*#*

*# Length can be negative*

function file\_range {

file =$1

start =$2

len=$3

case $len in

"end") my\_dd if= $file skip = $start || exit

1; return 0;;

" start ") my\_dd if= $file count = $start ||

exit 1; return 0;;

97

"") echo "len ␣was ␣ empty ␣( file :␣$file ,␣ start

:␣$start ,␣len␣ $len ).␣ Correct ␣ format ␣<

filename >␣<byte ␣start >␣<length |’start ’|’

end ’>" 1 >&2; exit 1;;

\*) ;;

esac

if [[ $len -gt 0 ]]; then

*# Dump to stdout*

my\_dd if= $file skip = $start count = $len ||

exit 1

else

skip =$(( $start + ( $len )))

len=$((- ( $len )))

if [[ $skip -lt 0 ]]; then

skip =0

len= $start

fi

*# Dump to stdout*

my\_dd if= $file skip = $skip count = $len ||

exit 1

fi

}

function backwards {

tac -b | rev

}

function byte\_offset {

grep -b -o -m 1 -F "$1" | cut -d : -f1

}

*# Throw everything but the page in stdout*

*#*

*# neg\_xml\_page " Barack Obama "*

function neg\_xml\_page {

term ="<title >$1 </ title >"

title\_offset =$(cat $ORIGINAL\_XML |

byte\_offset " $term ")

echo -e "\n\ tMethod :␣$2( blank ␣is␣ok)" 1 >&2

echo -e "\ tsearch ␣ term :␣ $term " 1 >&2

echo -e "\ tfile :␣ $ORIGINAL\_XML " 1 >&2

echo -e "\ ttitle ␣ offset :␣ $title\_offset " 1 >&2

98

*# Fail the term is invalid*

if [ -z " $title\_offset " ]; then

echo " Found ␣’$title\_offset ’␣Grep - ing␣(cat␣␣

$ORIGINAL\_XML ␣|␣ grep ␣-b␣-m␣1␣-F␣\" $term \

"␣|␣cut ␣-d:␣-f1)" 1 >&2

exit 1

fi

to\_page\_start =$(($( file\_range $ORIGINAL\_XML

$title\_offset -1000 | backwards |

byte\_offset "$( echo ␣ ’<page >’␣|␣rev)")+7) )

echo -e "\tto ␣ page ␣ start ␣( relative ):␣

$to\_page\_start " 1 >&2

file\_range $ORIGINAL\_XML $title\_offset end |

byte\_offset " </page >" >&2

echo $(($( file\_range $ORIGINAL\_XML

$title\_offset end | byte\_offset " </page >")

+7) ) >&2

to\_page\_end =$(($( file\_range $ORIGINAL\_XML

$title\_offset end | byte\_offset " </page >")

+7) ) *# len (’</page >’) == 7*

echo -e "\tto ␣ page ␣ end␣( relative ):␣

$to\_page\_end " 1 >&2

page\_start =$(( $title\_offset - $to\_page\_start

+1 ))

echo -e "\ tpage ␣ start :␣ $page\_start " 1 >&2

page\_end =$(( $title\_offset + $to\_page\_end ))

echo -e "\ tpage ␣end:␣ $page\_end " 1 >&2

echo -e "\ tbytes ␣to␣ copy :␣$(($(du␣-b␣

$ORIGINAL\_XML ␣|␣cut␣-f1)␣-␣ $page\_start ␣+␣

$page\_end ))" 1 >&2

echo " Going ␣to␣ copy ␣ $page\_start ␣ bytes " 1 >&2

file\_range $ORIGINAL\_XML $page\_start start

echo " Finished ␣the ␣ first ␣ half ␣up␣to␣

$page\_start ,␣$((␣$(du␣-b␣ $ORIGINAL\_XML ␣|␣

cut ␣-f␣1)␣-␣ $page\_end ␣))␣to␣go" 1 >&2

file\_range $ORIGINAL\_XML $page\_end end

echo " Finished ␣the ␣ whole ␣ thing ." 1 >&2

}

99

*# Put stdin betwinn mediawiki tags and into*

*stdout*

function mediawiki\_xml {

( head -1 $ORIGINAL\_XML ; sed -n "/< siteinfo

>/ ,/ <\/ siteinfo >/p;/ <\/ siteinfo >/q"

$ORIGINAL\_XML ; cat - ; tail -1

$ORIGINAL\_XML )

}

*# 1: XML File*

*# 2: Article*

*# 3: Method ( leave blank )*

*# Assert that the file is there and is not empty*

fsize =$(du -b $ORIGINAL\_XML | cut -f1)

if [[ 0 -eq $fsize ]]; then

echo " ERROR :␣ empty ␣xml␣ file ␣ $ORIGINAL\_XML "

1 >&2

exit 1

fi

echo " Will ␣ remove ␣ article ␣’$2 ’␣ from ␣ file ␣$1␣( size

:␣ $fsize )" 1 >&2

if ! neg\_xml\_page "$2" "$3"; then

ret=$?

echo "XML␣ parsing ␣ script ␣ failed " 1 >&2

exit $ret ;

fi

100

Part V

Related CSAIL projects

101

Part VI

Conclusion

102