Spams

- Από νωρίς έγινε αντιληπτό πως οι μηχανές αναζήτησης είναι ένα μέσο που συνδέει τους διαφημιστές με τους πιθανούς αγοραστές
- Οι πωλητές έχουν ισχυρό ενδιαφέρον να δημιουργήσουν σελίδες οπυ εμφανίζονται ψηλά στα rankings
- Μια σελίδα που έχει πάρα πολλές αναφορές σε κάποιους όρους θα εμφανιστεί ψηλά αν υιοθετηθεί η tf μετρική
- Αυτό οδηγεί στα spams
- Πρόκειται για την χειραγώγηση του περιεχομένου των σελίδων

Spams

- Κάποιοι πιο προχωρημένοι spammers χρωματίζουν αυτούς τους όρους με το ίδιο χρώμα με το background
- Παρά το γεγονός ότι αυτοί οι όροι δεν φαίνονται στους χρήστες τυγχάνουν επεξεργασίας από τις μηχανές αναζήτησης
- Άλλη τεχνική είναι το cloaking
- Ο server του spammer επιστρέφει διαφορετικό περιεχόμενο ανάλογα με τον δέχεται αίτημα από μια μηχανή αναζήτησης ή τους χρήστες
- Άλλη τεχνική είναι η χρήση μιας doorway σελίδας
- Περιλαμβάνει περιεχόμενο και μεταδεδομένα που έχουν επιλεχθεί προσεκτικά ώστε να αυξάνουν το ranking

Spams

- Όταν ένας φυλλομετρητής αιτείται την doorway σελίδα ανακατευθύνεται σε μια σελίδα που περιλαμβάνει περιεχόμενο με πιο εμπορική χρήση
- Το spamming έχει οικονομικές επιπτώσεις και έχει δημιουργηθεί μια περιοχή που ονομάζεται Search Engines Optimization – SEO
- Οι μηχανές αναζήτησης εξελίχθηκαν και ενσωματώνουν spam detection κώδικες
- Μια τεχνική είναι η link analysis
- Η πρώτη μηχανή που τη χρησιμοποίησε ήταν η Google

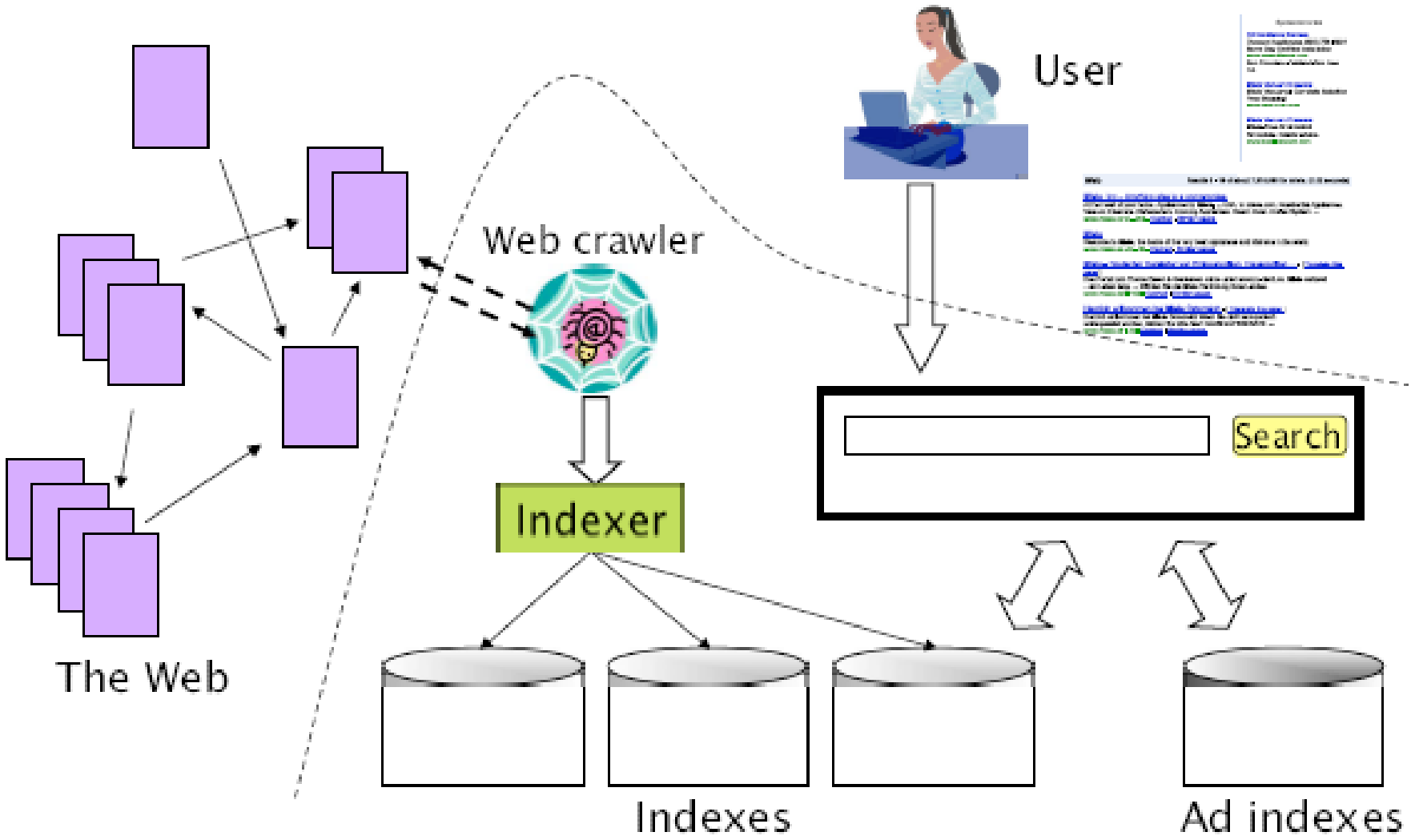
Ερωτήματα

- Σχετικά με τα ερωτήματα των χρηστών αυτά μπορούν να κατηγοριοποιηθούν ως ακολούθως:
  - Informational: στοχεύουν στην αναζήτηση γενικής πληροφορίας σε ένα θέμα – τυπικά, δεν είναι μόνο μια σελίδα που περιέχει το αποτέλεσμα
  - Navigational: αναζητούν τη σελίδα μιας οντότητας (εταιρεία, κ.λπ.) που θέλει ο χρήστης – προφανώς ο χρήστης αναμένει το σωστό αποτέλεσμα στην πρώτη θέση της λίστας
  - Transactional: σχετίζεται με την αναζήτηση και ολοκλήρωση μιας συναλλαγής π.χ. μια αγορά – η μηχανή αναζήτησης πρέπει να επιστρέψει τις σελίδες που δίνουν τη δυνατότητα της συναλλαγής π.χ. φόρμες

Ερωτήματα

- Η διάκριση των κατηγοριών των ερωτημάτων αποτελεί μια πρόκληση
- Αρχικά διότι η κατηγορία θα μας καθοδηγήσει στο αλγοριθμικό κομμάτι με το οποίο θα επιστραφούν τα αποτελέσματα
- Επίσης, η κατηγορία θα καθορίσει την καταλληλότητα του ερωτήματος για συγκεκριμένα αποτελέσματα

Ερωτήματα



Μέγεθος Ευρετηρίου

- Κάποιες σελίδες περιέχουν περισσότερη πληροφορία από ότι κάποιες άλλες
- Επίσης, είναι δύσκολο να ορίσουμε το ποσοστό του Web που είναι indexed από μια μηχανή αναζήτησης αφού υπάρχουν πάρα πολλές δυναμικές σελίδες
- Πολλές σελίδες ‘παγιδεύουν’ τις μηχανές αναζήτησης ώστε να τοποθετήσουν στα ευρετήρια όσο το δυνατόν περισσότερες σελίδες του Δικτυακού τόπου
- Γενικά, δεν μπορούμε να έχουμε συγκεκριμένο αριθμό που να απεικονίζει το μέγεθος των ευρετηρίων

Μέγεθος Ευρετηρίου

- Οι λόγοι είναι:
  - Αρκετές φορές οι μηχανές αναζήτησης επιστρέφουν αποτελέσματα που δεν έχουν γίνει indexed – γενικά μόνο οι πρώτες χιλιάδες λέξεων περνούν από το indexing
  - Οι μηχανές αναζήτησης οργανώνουν τα ευρετήρια σε πολλαπλά επίπεδα και δεν υιοθετούνται όλα σε κάθε αναζήτηση

Web Crawling

Εισαγωγή

- Το Web crawling αφορά τη διαδικασία μέσω της οποίας συλλέγουμε ιστοσελίδες από το Web
- Στη συνέχεια δημιουργούμε τα ευρετήρια που υποστηρίζουν τις μηχανές αναζήτησης
- Ο στόχος είναι η γρήγορη συλλογή όσων περισσότερων και πιο χρήσιμων ιστοσελίδων μπορούμε
- Επίσης, συλλέγουμε και τη δομή των υπερσυνδέσμων
- Άλλη ονομασία είναι spider

Εισαγωγή

- Τα στοιχεία που πρέπει ένας Web crawler να υποστηρίζει:
  - **Robustness**: το Web περιέχει servers που δημιουργούν spider traps ώστε να τους καθοδηγήσουν στις σελίδες που θέλουν – πολλές φορές τους παραπέμπουν σε ένα μεγάλο αριθμό σελίδων – οι crawlers πρέπει να αποφεύγουν αυτές τις παγίδες
  - **Politeness**: οι servers έχουν συγκεκριμένες πολιτικές με τις οποίες ορίζουν το ρυθμό με τον οποίο τους επισκέπτονται οι crawlers

Εισαγωγή

- Τα στοιχεία που μπορεί (θα μπορούσε) ένας Web crawler να υποστηρίζει:
  - **Distributed**: θα πρέπει να μπορεί να εκτελεστεί σε πολλές μηχανές
  - **Scalable**: θα πρέπει να μπορεί να επεκταθεί το εύρος ζώνης και το πλήθος των μηχανών στις οποίες εκτελείται
  - **Performance and Efficiency**: θα πρέπει να κάνει σωστή χρήση των υπολογιστικών πόρων
  - **Quality**: θα πρέπει να συλλέγει μόνο τις πιο χρήσιμες σελίδες
  - **Freshness**: πρέπει να προσπελαύνει τις πιο πρόσφατες εκδόσεις των σελίδων
  - **Extensible**: θα πρέπει να επεκτείνονται εύκολα – νέες μορφές δεδομένων, νέα πρωτόκολλα

Crawling

- Η βασική λειτουργία οποιουδήποτε crawler εκκινεί από ένα σύνολο σελίδων που ονομάζεται seed set
- Παίρνει ένα URL από αυτό το σύνολο και ανακτά την ιστοσελίδα
- Η ιστοσελίδα τυγχάνει επεξεργασίας για να εξαχθεί το κείμενο και οι σύνδεσμοι
- Το κείμενο στέλνεται στο λογισμικό που εξάγει το ευρετήριο
- Οι σύνδεσμοι μπαίνουν στο URL frontier που αποτελείται από τα URLs των οποίων οι σελίδες πρόκειται να ανακτηθούν
- Αρχικά, το frontier set περιλαμβάνει μόνο το seed set
- Όσες σελίδες τυγχάνουν επεξεργασίας, διαγράφονται από τα προαναφερόμενα σύνολα

Crawling

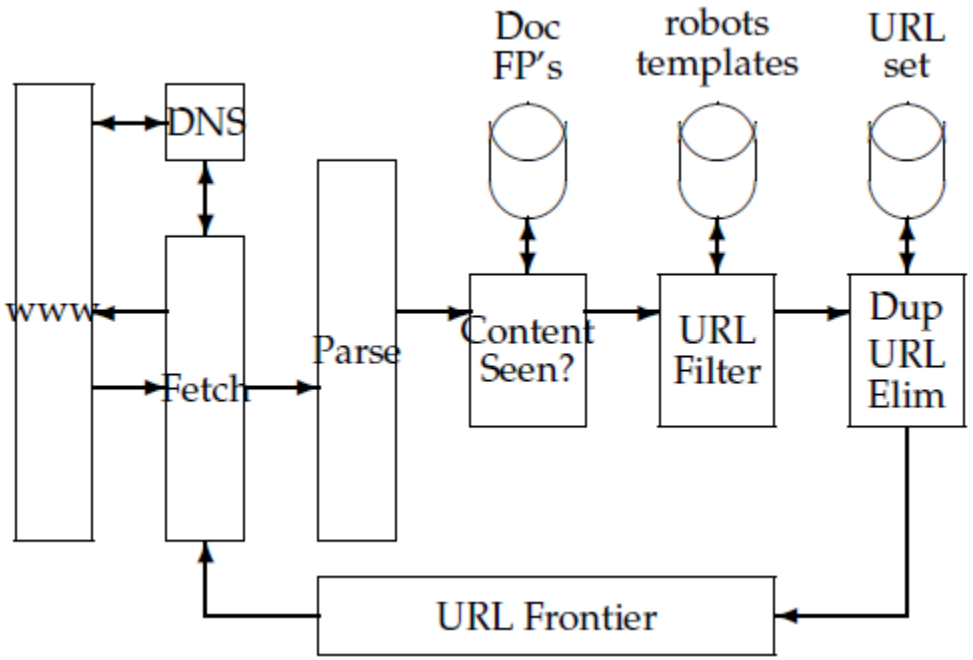
- Ουσιαστικά πρόκειται για τη διάσχιση του Web γράφου
- Στοτ συνεχόμενο crawling κάθε URL μετά την επεξεργασία του μπαί΄νει στο τέλος του frontier
- Στη συνέχεια θα αναλυθεί το μοντέλο του Mercator crawler που αποτελεί τη βάση ενός συνόλου crawlers
- Ακολουθείται ένα σχήμα multi-thread ώστε να αυξηθεί ο αριθμός των σελίδων που ανακτώνται
- Η επεξεργασία ενός δισεκατομυρίου σελίδων σε ένα μήνα απαιτεί την επεξεργασία εκατοντάδων σελίδων το δευτερολεπτο

Crawling

- Κάποια βασικά σημεία:
  - Μόνο μια σύνδεση πρέπει να είναι ανοιχτή με κάθε host κάθε φορά
  - Ένας χρόνος αναμονής της τάξης των δευτερολέπτων πρέπει να υπάρχει ανάμεσα σε συνεχόμενες αιτήσεις προς ένα host
  - Η υιοθέτηση των κανόνων των hosts (politeness) πρέπει να ακολουθείται πιστά

Αρχιτεκτονική

- Τμήματα:
  - Το URL frontier σύνολο
  - Το DNS resolution module – καθορίζει τον εξυπηρετητή από όπου θα ανακτήσουμε το URL
  - Μια μοινάδα ανάκτησης του περιεχομένου που υιοθετεί το http πρωτόκολλο
  - Μια μονάδα επεξεργασίας που εξάγει το κείμενο
  - Μια μονάδα που εξαλείφει τα διπλότυπα των ανακτήσεων στα URLs



Διαδικασία

- Μπορεί να εκτελεστεί από οπουδήποτε και υιοθετεί από ένα μέχρι εκατοντάδες νήματα
- Κάθε νήμα υλοποιεί την αρχιτεκτονική που είδαμε

Διαδικασία

- Εκκινούμε παίρνοντας το URL από το frontier set
- Ανακτούμε την αντίστοιχη σελίδα
- Η σελίδα αποθηκεύεται σε ένα χώρο προσωρινής αποθήκευσης
- Εξάγουμε το κείμενο
- Εξάγουμε τους συνδέσμους
- Ελέγχουμε τους συνδέσμους για να δούμε αν μπορεί να αποθηκευτεί στο URL frontier

Διαδικασία

- Κάθε νήμα ελέγχει αν μια σελίδα με το ίδιο περιεχόμενο έχει ήδη απαντηθεί σε ένα άλλο URL
- Ο πιο απλός τρόπος υλοποίησης είναι να υιοθετήσουμε ένα αποτύπωμα (fingerprint) – π.χ., checksum
- Στη συνέχεια ένα URL filter καθορίζει αν το URL που εξάγουμε μπορεί να απορριφθεί με βάση κάποια τεστ
- Μπορεί να θέλουμε να αποφύγουμε κάποια URLs
- Παράδειγμα:
  - Απορρίπτουμε όλα τα .net domains

Διαδικασία

- Μπορεί το τεστ να είναι inclusive και όχι exclusive
- Αρκετοί hosts τοποθετούν κάποια τμήματα σελίδων έξω από τα όρια των crawlers
- Ονομάζεται Robots Exclusion Protocol
- Τοποθετούν το αρχείο robots.txt στο root του URL
- Παράδειγμα:
  - Απορρίπτουμε όλες τις προσβάσεις σε οποιοδήποτε URL βρίσκεται κάτω από το /yoursite/temp/ εκτός από το searchengine

```
User-agent: *
Disallow: /yoursite/temp/

User-agent: searchengine
Disallow:
```

Διαδικασία

- Το αρχείο robots.txt πρέπει να ανακτηθεί για να διαπιστωθεί αν το URL περνάει από τους περιορισμούς
- Αν ναι, τότε το URL προστίθεται στο URL frontier
- Μπορούμε να υιοθετήσουμε και caches
- Η χρήση της cache βοηθάει αφού πολλαπλά links θα μας στέλνουν στον ίδιο host οπότε πρέπει να ελεγχθούν αν περνούν από τους περιορισμούς
- Δυστυχώς, ένα URL μπορεί να είναι στο frontier set για αρκετό καιρό π.χ. Μέρες, εβδομάδες
- Οι σύνδεσμοι σε αυτό το URL μπορεί να μην μπορούν να ελεγχθούν με τη βοήθεια της cache αφού το robots.txt μπορεί να έχει αλλάξει
- Ο έλεγχος με το robots.txt πρέπει να γίνει πριν την ανάκτηση της σελίδας

Διαδικασία

- Έπειτα, το URL πρέπει να κανονικοποιηθεί (normalized URL)
- Ο λόγος είναι ότι συχνά η κωδικοποίηση HTML ενός συνδέσμου δείχνει το στόχο σχετικά με την ίδια τη σελίδα
- Παράδειγμα:
  - en.wikipedia.org/wiki/Main_Page
  - Disclaimers
  - http://en.wikipedia.org/wiki/Wikipedia:General_disclaimer

Διαδικασία

- Τέλος, το URL ελέγχεται για διπλότυπα
- Αν το URL είναι ήδη στο frontier set και το έχουμε ήδη ανακτήσει τότε δεν το ξαναναποθηκεύουμε
- Όταν το URL μπαίνει στο frontier set του ανατίθεται μια προτεραιότητα

Παραδείγματα

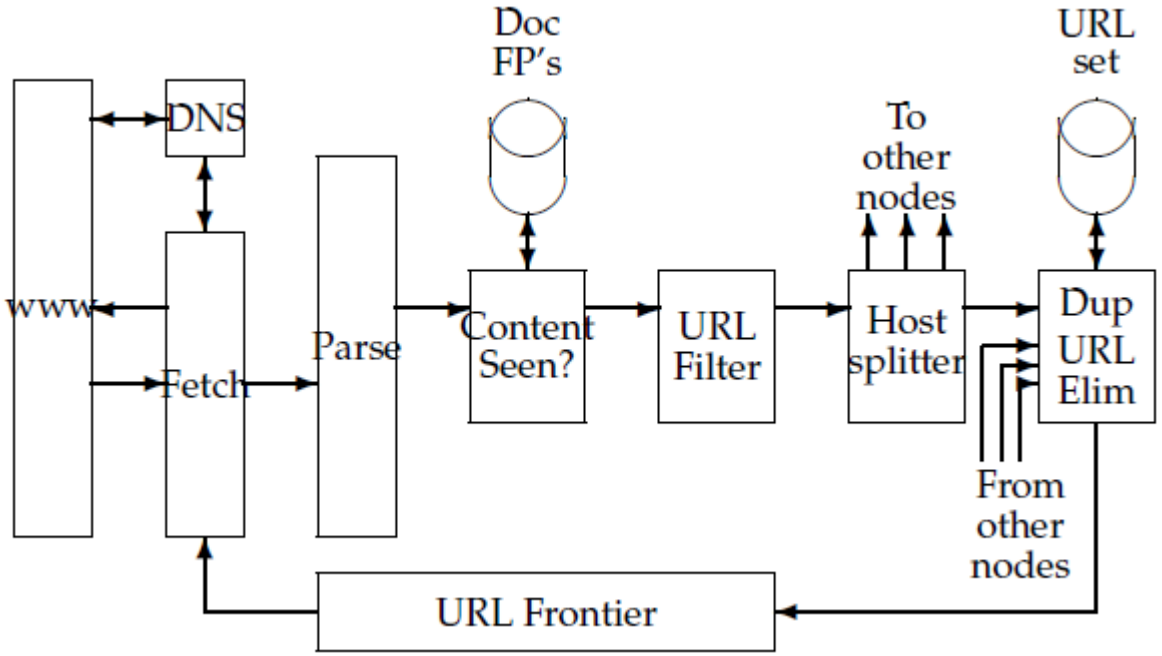
- http://www.makeuseof.com/tag/build-basic-web-crawler-pull-information-website-2/
- https://subinsb.com/how-to-create-a-simple-web-crawler-in-php/
- https://hotexamples.com/examples/-/Spider/-/php-spider-class-examples.html

Distributed Crawling

- Τα νήματα μπορεί να εκτελούνται σε διαφορετικούς κόμβους
- Αυτό βοηθά στο scalability
- Επίσης μπορεί να βοηθήσει στη γεωγραφική κατανομή των σελίδων
- Η τμηματοποίηση των crawlers μπορεί να γίνει με μια συνάρτηση κατακερματισμού ή με κάποια άλλη στρατηγική
- Παράδειγμα:
  - Τοποθετούμε ένα crawler στην Ευρώπη που να χειρίζεται τα Ευρωπαικά domains
  - Όμως η δρομολόγηση των πακέτων δεν απεικονίζει πάντα τη γεωγραφική κατανομή

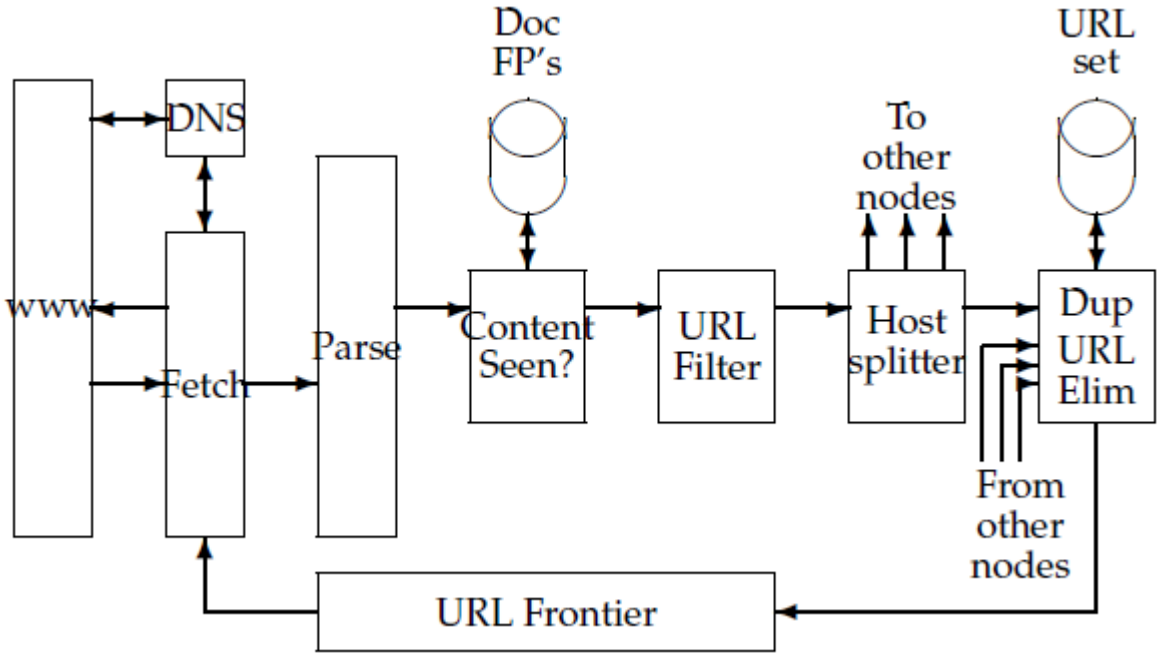
Distributed Crawling

- Αντιγράφουμε την προηγούμενη διαδικασία σε κάθε κόμβο και στη συνέχεια εφαρμόζουμε ένα host splitter
- Ο host splitter 'στέλνει' κάθε URL στον υπεύθυνο crawler



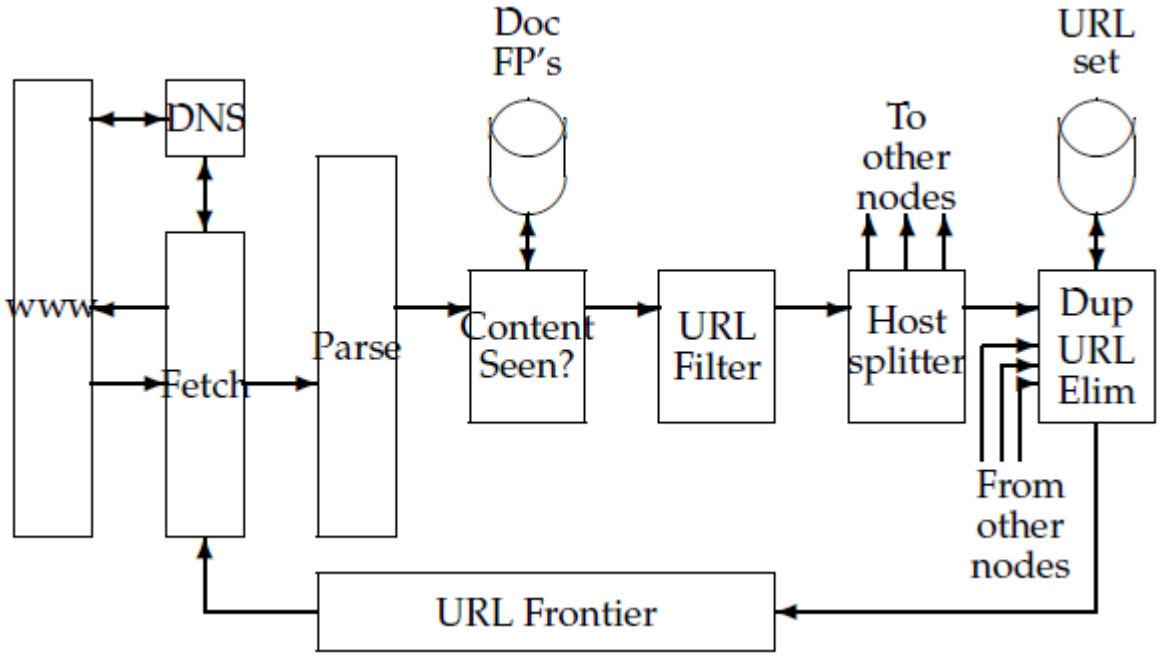
Distributed Crawling

- Όμως το Content seen module είναι πιο περίπλοκο
- Τα fingerprints δεν μπορούν να τμηματοποιηθούν βασιζόμενοι στο όνομα του host
- Μπορεί το ίδιο περιεχόμενο να παρουσιαστεί σε διαφορετικούς εξυπηρετητές
- Συνήθως, βασιζόμαστε σε remote procedure calls για τα fingerprints



Distributed Crawling

- Δεν μπορούμε να αποθηκεύσουμε σε cache τα πιο δημοφιλή fingerprints
- Τα έγγραφα αλλάζουν στο χρόνο και πρέπει να μπορούμε να διαγράψουμε τα παλιά fingerprints
- Άρα, πρέπει να αποθηκεύουμε τα fingerprints μαζί με τα URLs



DNS Resolution

- Το DNS resolution είναι η αντιστοίχιση ενός URL σε μια μοναδική IP ενός server
- Ένας crawler επικοινωνεί με ένα DNS server ώστε να λάβει την IP διεύθυνση
- Όταν το URL είναι περίπλοκο (πολλαπλά τμήματα) τότε το αντίστοιχο module του crawler ανακτά το domain

DNS Resolution

- Το DNS resolution προκαλεί συμφόρηση στο crawling
- Μια αίτηση DNS μπορεί να περιλαμβάνει πολλές αιτήσεις που να απαιτούν δευτερόλεπτα για να προκύψει μια απάντηση
- Προφανώς, θα επιβαρύνει το χρόνο ανάκτησης των σελίδων
- Μια λύση είναι το caching
- Για τα URLs για τα οποία έχουμε πρόσφατες DNS αιτήσεις μπορούμε να αποθηκεύσουμε την IP

DNS Resolution

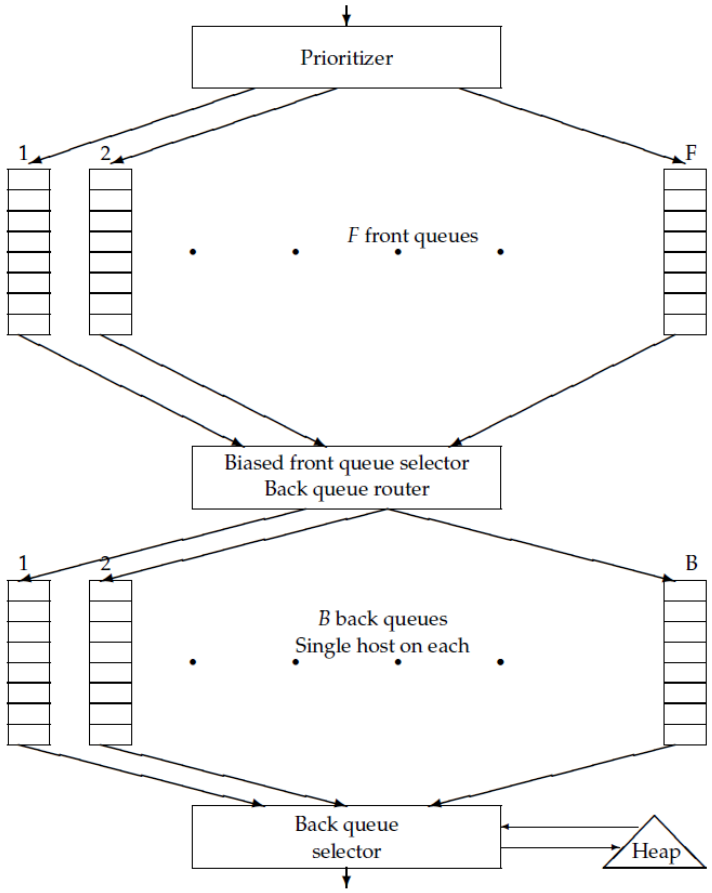
- Μια άλλη δυσκολία είναι οι σύγχρονες κλήσεις
- Τα νήματα μπλοκάρονται μέχρι να έρθει η απάντηση στο πρώτο αίτημα
- Για να ξεπεράσουμε αυτό το πρόβλημα, αρκετοί crawlers υλοποιούν τους δικούς τους DNS resolvers
- Το νήμα resolver στέλνει αίτημα στον DNS server
- Επανακτά όταν δεχθεί μήνυμα από τον server ή από κάποιο άλλο νήμα ή περάσει ένα όριο χρόνου
- Ένα άλλο DNS νήμα ακούει στο DNS port (53)
- Όταν ληφθεί απάντηση στέλνει μήνυμα στον resolver
- Όταν περνά το όριο χρόνου ξαναεπιχειρείται η αποστολή της αίτησης

URL Frontier

- Διατηρείται μια λίστα URLs και ανακτάται με κάποια σειρά
- Ακολουθούνται δύο προσεγγίσεις:
  - Σελίδες υψηλής ποιότητας που αλλάζουν συχνά πρέπει να έχουν ηψηλότερη προτεραιότητα – συνεπώς, η προτεραιότητα μια σελίδας μπορεί να είναι συνάρτηση του ρυθμού αλλαγής και της ποιότητας της – ο συνδυασμός είναι απαραίτητος αν σκεφτούμε πως οι spam σελίδες αλλάζουν συχνά
  - Πρέπει να αποφεύγεται η επαναλαμβανόμενη ανάκτηση μέσα σε σύντομο χρονικό διάστημα – πολλά URLs δείχνουν σε 'τοπικές' σελίδες – αν τις υιοθετήσουμε με μια απλή ουρά θα προκληθούν bursts αιτήσεων προς ένα host – μια λύση είναι να εισάγουμε ένα κενό ανάμεσα σε συνεχόμενες αιτήσεις

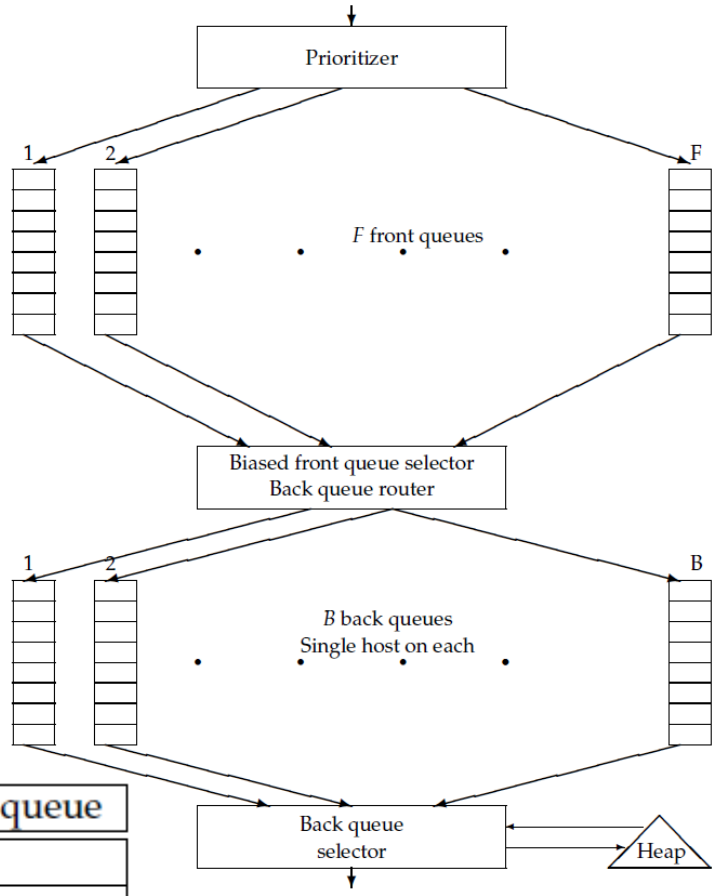
URL Frontier

- Έχουμε ένα σύνολο από front queues και ένα σύνολο από back queues
- Όλες είναι ουρές FIFO
- Οι front queues υλοποιούν τις προτεραιότητες ενώ οι back το politeness
- Ένας prioritizer αναθέτει μια προτεραιότητα στο URL ανάμεσα στο 1 και το F (παίρνοντας υπόψιν το αν έχει αλλάξει η σελίδα)
- Παράδειγμα: αν μια σελίδα αλλάζει συχνά παίρνει μεγαλύτερη προτεραιότητα



URL Frontier

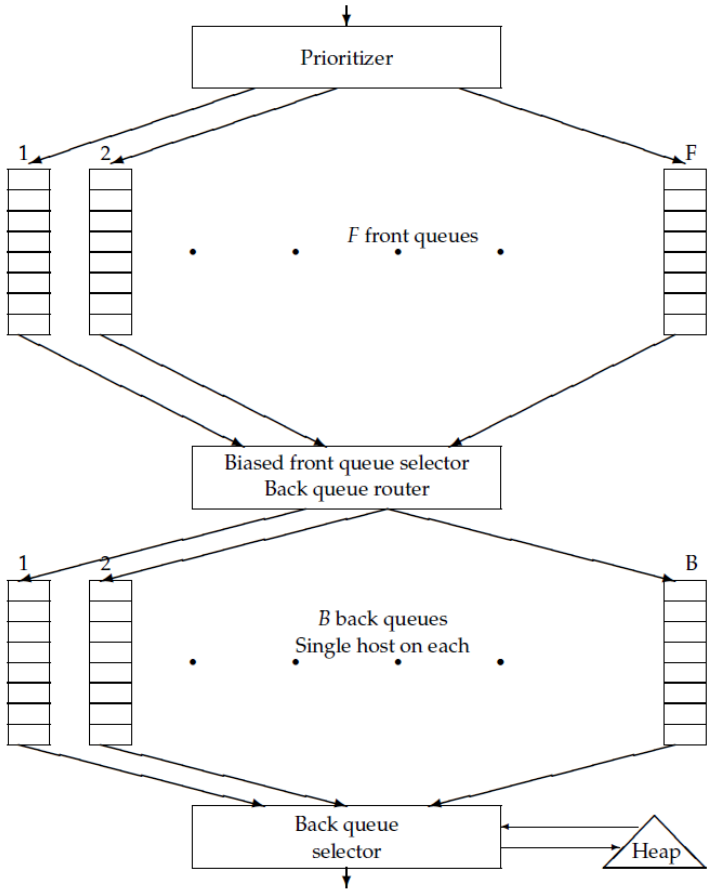
- Με βάση την προτεραιότητα, το URL μπαίνει στην αντίστοιχη ουρά
- Κάθε back queue δεν είναι άδεια όταν λειτουργεί ο crawler και περιέχει URLs από ένα host
- Ένας βοηθητικός πίνακας T διατηρεί την αντιστοίχιση ανάμεσα σε hosts και back queues
- Όταν μια back queue αδειάζει, τότε τροφοδοτείται με URLs από τις front queues και ενημερώνεται ο T
- Τέλος διατηρείται και ένας σωρός με μια καταχώρηση για κάθε back queue
- Καταχώρηση: χρόνος που μπορούμε να επικοινωνήσουμε με ένα host



Host	Back queue
stanford.edu	23
microsoft.com	47
acm.org	12

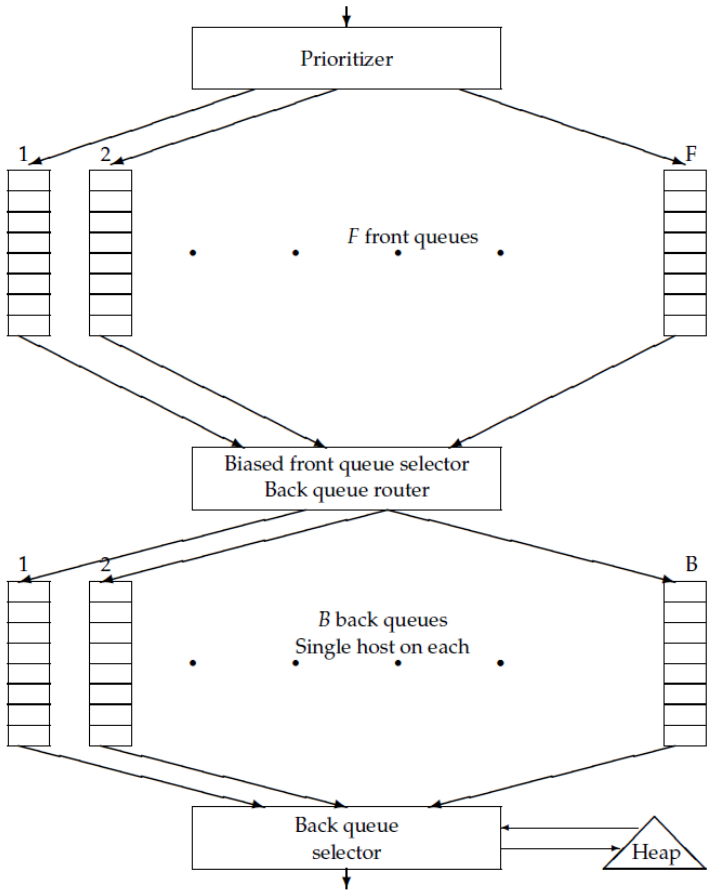
URL Frontier

- Ένα νήμα που ζητά URL από το frontier εξάγει τη ρίζα του σωρού και περιμένει για όσο χρόνο αντιστοιχεί
- Μετά την ανάκτηση, το νήμα ελέγχει αν η ουρά είναι άδεια
- Αν ναι, εξάγει ένα URL από μια front queue (συνήθως επιλέγεται τυχαία)
- Η επιλογή ναι μεν είναι τυχαία αλλά με προτίμηση στα URLs με μεγάλη προτεραιότητα
- Εξετάζουμε το επιλεγμένο URL για το αν υπάρχει back queue για τον ίδιο host



URL Frontier

- Αν ναι, τότε την εισάγουμε στην ουρά και πάμε πίσω στις front queues και εξάγουμε και επόμενο URL ώστε να διαπιστώσουμε αν μπορεί να μπει στην άδεια αρχική back queue
- Για το URL που μπαίνει στις back queues κάνουμε αντίστοιχη εισαγωγή στο σωρό με τους χρόνους



Connectivity Servers

- Οι μηχανές αναζήτησης χρειάζονται ένα connectivity server που να υποστηρίζουν γρήγορα connectivity queries
- Τυπικά ερωτήματα αυτής της μορφής είναι:
  - Ποια URL συνδέονται με ένα URL?
  - Με ποια URLs συνδέεται ένα συγκεκριμένο URL?
- Θέλουμε να αποθηκεύσουμε αντιστοιχίσεις URL-out links, URL-in links
- Εφαρμογές αποτελούν: link analysis, web graph analysis, crawl optimization, crawl control

Connectivity Servers

- Ας υποθέσουμε πως κάθε σελίδα αντιστοιχεί σε ένα μοναδικό αριθμό
- Χτίζουμε ένα πίνακα γειτνίασης (adjacency table): κάθε γραμμή αντιστοιχεί σε μια σελίδα (ταξινομημένες ως προς το μοναδικό αριθμό)
- Κάθε γραμμή περιέχει μια ταξινομημένη λίστα ακεραίων που αντιστοιχεί σε συνδέσμους των σελίδων
- Συνεπώς, μπορούμε να απαντήσουμε στα προηγούμενα ερωτήματα
- Όμως, όσο περισσότεροι είναι οι σύνδεσμοι και οι σελίδες τόσο μεγαλύτερος είναι ο πίνακας

Connectivity Servers

- Υιοθετούμε τεχνικές για τη μείωση του μεγέθους του πίνακα
  - Ομοιότητα ανάμεσα στις λίστες: πολλές γραμμές έχουν κοινές καταχωρήσεις στις λίστες τους
  - Τοπικότητα: πολλοί σύνδεσμοι σε μια σελίδα οδηγούν σε γειτονικές σελίδες
  - Χρησιμοποιούμε κωδικοποίηση κενού: αντί να αποθηκεύσουμε τον προορισμό κάθε συνδέσμου αποθηκεύουμε τη διαφορά από την προηγούμενη σελίδα

Connectivity Servers

- Σε μια λεξικογραφική σειρά όλων των URLs χειριζόμαστε τα URLs σαν αλφαριθμητικά και τα ταξινομούμε
- Για μια πραγματική λεξικογραφική σειρά το domain πρέπει να αντιστραφεί π.χ. www.uth.gr -> gr.uth.www, χωρίς να είναι απαραίτητο
- Σε κάθε URL αναθέτουμε τη θέση του στη σειρά

```
1: www.stanford.edu/alchemy
2: www.stanford.edu/biology
3: www.stanford.edu/biology/plant
4: www.stanford.edu/biology/plant/copyright
5: www.stanford.edu/biology/plant/people
6: www.stanford.edu/chemistry

 1: 1, 2, 4, 8, 16, 32, 64
 2: 1, 4, 9, 16, 25, 36, 49, 64
 3: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144
 4: 1, 4, 8, 16, 25, 36, 49, 64
```

Connectivity Servers

- Κάθε σελίδα έχει ένα template με ένα σύνολο συνδέσμων από κάθε σελίδα σε ένα σταθερό σύνολο σελίδων στον ίδιο δικτυακό τόπο
- Οι γραμμές θα έχουν πολλές καταχωρήσεις κοινές
- Επίσης, με βάση τη λεξικογραφική σειρά, είναι πιθανό οι σελίδες από ένα δικτυακό τόπο να εμφανίζονται σαν συνεχόμενες θέσεις στον πίνακα

```
1: www.stanford.edu/alchemy
2: www.stanford.edu/biology
3: www.stanford.edu/biology/plant
4: www.stanford.edu/biology/plant/copyright
5: www.stanford.edu/biology/plant/people
6: www.stanford.edu/chemistry
```

```
1: 1, 2, 4, 8, 16, 32, 64
2: 1, 4, 9, 16, 25, 36, 49, 64
3: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144
4: 1, 4, 8, 16, 25, 36, 49, 64
```

Connectivity Servers

- Στρατηγική
  - Κατεβαίνουμε στον πίνακα και κωδικοποιούμε κάθε γραμμή με βάση τις 7 προηγούμενες γραμμές
  - Παράδειγμα: κωδικοποιούμε την 4η ως όμοια με τη γραμμή σε offset 2 με το 9 να αντικαθίσταται από το 8
  - Λαμβάνοντας τις 7 προηγούμενες γραμμές απαιτεί 3 bits για το offset
  - Το offset θα έχει μια μικρή τιμή και έτσι δεν χρειάζεται να κάνουμε μια αναζήτηση με υψηλό κόστος

```
1: www.stanford.edu/alchemy
2: www.stanford.edu/biology
3: www.stanford.edu/biology/plant
4: www.stanford.edu/biology/plant/copyright
5: www.stanford.edu/biology/plant/people
6: www.stanford.edu/chemistry

 1: 1, 2, 4, 8, 16, 32, 64
 2: 1, 4, 9, 16, 25, 36, 49, 64
 3: 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144
 4: 1, 4, 8, 16, 25, 36, 49, 64
```

Connectivity Servers

- Αν καμμία από τις προηγούμενες γραμμές δεν είναι κατάλληλη τότε εκφράζουμε τη γραμμή σαν να εκκινεί από το κενό σύνολο
- Αυτό συμβαίνει όταν μετακινούμαστε σε διαφορετικούς δικτυακούς τόπους

Connectivity Servers

- Για την υποστήριξη των ερωτημάτων:
  - Αρχικά κάνουμε μια αναζήτηση από το URL προς τον αριθμό της γραμμής (π.χ. Με μια συνάρτηση κατακερματισμού)
  - Ανακατασκεύαζουμε τη γραμμή αφού μπορεί να έχει αποθηκευθεί με αναφορές σε άλλες γραμμές
  - Ακολουθούμε τα offsets
  - Για να μην έχουμε συνεχόμενες αναφορές σε προηγούμενες γραμμές, κατά την κατασκευή όταν εξετάζουμε τις προηγούμενες 7 γραμμές απαιτούμε ένα όριο ομοιότητας
  - Πρέπει οι δύο γραμμές που συγκρίνονται να έχουν μεγαλύτερη ομοιότητα από το όριο
  - Χρειάζεται προσοχή στο όριο: μεγάλο όριο -> πολλές σελίδες αποθηκεύονται χωρίς αναφορά σε άλλες, μικρό όριο-> πολλές ανακατευθύνσεις κατά την εκτέλεση του ερωτήματος

Link Analysis

Εισαγωγή

- Η ανάλυση των υπερσυνδέσμων (link analysis) καθώς και η δομή του Web αποτελούν μια βάση για τον υπολογισμό ενός σκορ των ιστοσελίδων
- Έχει τις βάσεις της στην ανάλυση των αναφορών (citations)
- Όπως οι αναφορές αναδεικνύουν τη σημασία ενός συγγραφέα έτσι και η ανάλυση των συνδέσμων αναδεικνύει τη σημασία των σελίδων
- Φυσικά, δεν σημαίνει πως όλοι οι σύνδεσμοι έχουν την ίδια σημασία για μια σελίδα
- Για παράδειγμα, κάποιος μπορεί να φτιάξει πολλές σελίδες που να δείχνουν σε μια άλλη ώστε να αποτυπώσει πολλούς συνδέσμους
- Αυτό το φαινόμενο ονομάζεται link spam

Το Web ως ένας Γράφος

- Έχουμε ήδη μιλήσει για το γράφο που μπορεί να δημιουργηθεί μέσω των συνδέσμων στο Διαδίκτυο
- Η ανάλυση των συνδέσμων βασίζεται σε δύο παραμέτρους:
  - Το anchor text που δείχνει σε μια σελίδα Β είναι μια καλή περιγραφή της σελίδας Β
  - Ο σύνδεσμος από μια σελίδα Α προς τη Β αναπαριστά μια έγκριση/επιδοκιμασία (endorsement) της Β από το δημιουργό της Α
    - Αυτό φυσικά δεν συμβαίνει πάντα – παράδειγμα: πολλές εταιρικές σελίδες έχουν στο τέλος τους συνδέσμους προς κάποιες σελίδες με copyrights
    - Οι αλγόριθμοι ανάλυσης των συνδέσμων πρέπει να ανακαλύπτουν αυτού του είδους τους συνδέσμους

Anchor Text

- Ας δούμε ένα παράδειγμα ενός υπερσυνδέσμου

  Journal of the ACM.
- Ο σύνδεσμος ‘δείχνει’ προς τη σελίδα http://www.acm.org/jacm/ και το κείμενο που εμφανίζεται είναι το *Journal of the ACM*
- Σε αυτό το παράδειγμα το anchor text περιγράφει τη σελίδα προς την οποία μας παραπέμπει ο σύνδεσμος
- Όμως, ο ίδιος ο σύνδεσμος μας περιγράφει τη σελίδα επίσης
- Σε αυτή την περίπτωση ίσως το κείμενο να πλεονάζει

Anchor Text

- Το Διαδίκτυο είναι γεμάτο από υπερσυνδέσμους προς π.χ. μια σελίδα Β που δεν περιλαμβάνει μια περιγραφή όπως το προηγούμενο παράδειγμα
- Σε πολλές περιπτώσεις είναι επιλογή των κατασκευαστών να ορίσουν το κείμενο καθώς και το όνομα των σελίδων
- Το φαινόμενο αυτό είναι ιδιαίτερα έντονο στις περιπτώσεις των εταιρικών σελίδων όπου λαμβάνονται υπόψιν και στρατηγικές μάρκετινγκ
- Παράδειγμα:
  - Η σελίδα https://www.ibm.com/gr-el/ της IBM δεν έχει αναφορές στον κώδικά της στη λέξη κλειδί ‘υπολογιστής’ ή ‘υπολογιστές’ παρά το γεγονός ότι κατασκευάζει υπολογιστές
  - Επίσης, η https://gr.yahoo.com/?p=us της Yahoo δεν περιλαμβάνει τη λέξη κλειδί portal

Anchor Text

- Με τα προηγούμενα παραδείγματα γίνεται αντιληπτό πως υπάρχει μια αντίφαση στο πως περιγράφονται οι σελίδες και το πως αντιλαμβάνονται οι χρήστες το περιεχόμενο των σελίδων
- Με αυτό το σκεπτικό υπάρχει προβληματισμός στη συσχέτιση των ερωτημάτων με τα περιεχόμενα των σελίδων
- Επιπλέον, αρκετές ιστοσελίδες περιέχουν εικόνες, video, γραφικά και ενσωματώνουν το κείμενο μέσα σε αυτά
- Όταν συμβαίνει αυτό, οι crawlers δεν μπορούν να ανακτήσουν αυτού του είδους το κείμενο

Anchor Text

- Όμως, τα σχετικά κείμενα μπορούμε να τα προσπελάσουμε από τα anchor texts που περιέχουν οι σελίδες που δείχνουν σε κάποιες άλλες
- Για παράδειγμα, είναι γεγονός ότι πολλοί υπερσύνδεσμοι περιλαμβάνουν anchor text με τη λέξη κλειδί ‘υπολογιστής’ και δείχνουν προς τη σελίδα http://www.ibm.com
- Έτσι, τα terms των anchor texts μπορούν να περιληφθούν στα ευρετήρια και να δείχνουν προς τη συγκεκριμένη σελίδα
- Τα postings για τον όρο ‘υπολογιστής’ θα περιλαμβάνουν το έγγραφο http://www.ibm.com

Anchor Text

- Όμοια με τους όρους των κειμένων, οι όροι που περιλαμβάνονται στα anchor texts χαρακτηρίζονται από βάρη με βάση τη συχνότητά τους
- Επίσης, υπάρχει και ένα πέναλτι για τους όρους που απαντώνται πολύ συχνά
- Το τελικό βάρος καθορίζεται από μεθόδους μηχανικής μάθησης όπως έχουμε ήδη δει

Anchor Text

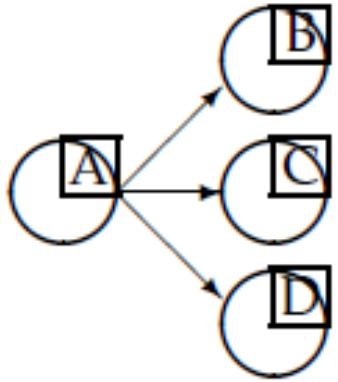
- Υπάρχουν όμως και κάποιες ‘παρενέργειες’:
  - Υπάρχει η πιθανότητα κάποιες σελίδες να χρησιμοποιούν μαζικά κάποιο κείμενο που μας παραπέμπει σε διαφορετική σελίδα στόχο
  - Η αναγνώριση αυτών των κειμένων είναι μια άλλη μορφή spam detection
  - Συχνά χρησιμοποιείται το παράθυρο (window) γύρω από το anchor text με τον ίδο τρόπο
  - Παράδειγμα:
    - there is good discussion about vehicles <a>here

PageRank

- Θα εστιάσουμε σε μεθόδους υπολογισμού βαρών και σκορ όπως εξάγονται από την ανάλυση των συνδέσμων
- Μια τεχνική είναι να αναθέσουμε ένα σκορ στο [0,1] σε κάθε κόμβο του Διαδικτυακού γράφου
- Η τιμή αυτή είναι γνωστή ως PageRank
- Η τιμή ενός κόμβου εξαρτάται από τη δομή των συνδέσμων στο γράφο
- Μόλις τεθεί ένα ερώτημα, η μηχανή αναζήτησης υπολογίζει ένα σύνθετο σκορ/τιμή που συνδυάζει πολλαπλά χαρακτηριστικά με το PageRank του κάθε κόμβου

PageRank

- Παράδειγμα:
  - Έστω ότι ένας χρήστης βρίσκεται στη σελίδα Α
  - Έχει τη δυνατότητα να πλοηγηθεί σε ένα από τους τρεις διαθέσιμους συνδέσμους
  - Αν θεωρήσουμε ένα τυχαίο ‘περίπατο’, τότε η επιλογή του κάθε συνδέσμου έχει πιθανότητα 1/3
  - Καθώς ο χρήστης συνεχίζει αυτόν τον τυχαίο περίπατο (random walk) πιθανόν να επισκεφθεί κάποιες σελίδες πιο συχνά από κάποιες άλλες
  - Μπορεί να υπάρχουν κόμβοι με πολλούς εισρχόμενους συνδέσμους από άλλους δημοφιλείς κόμβους



PageRank

- Ας υποθέσουμε πως στη σελίδα Α στην οποία βρίσκεται ο χρήστης δεν έχει εξερχόμενους συνδέσμους
- Για να χειριστούμε αυτή την κατάσταση υιοθετούμε την teleport λειτουργικότητα
- Σε αυτή ο χρήστης μεταπηδά από ένα κόμβο σε κάποιον άλλο
- Αυτό μπορεί να συμβεί όταν γράφει το URL αντι να χρησιμοποιήσει ένα σύνδεσμο από κάποια σελίδα
- Ο προορισμός θεωρείται πως προκύπτει ομοιόμορφα ανάμεσα σε όλους του πιθανούς κόμβους

PageRank

- Η λειτουργικότητα teleport χρησιμοποιείται για τον υπολογισμό του PageRank ως ακολούθως:
  - Όταν ένας κόμβος δεν έχει εξερχόμενους συνδέσμους, τότε επιλέγουμε ένα προορισμό ομοιόμορφα
  - Σε οποιοδήποτε κόμβο που έχει εξερχόμενους συνδέσμους, ο χρήστης επιλέγει την teleport λειτουργικότητα με πιθανότητα $0<\alpha<1$ και τον ‘κανονικό’ τυχαίο περίπατο με πιθανότητα $1-\alpha$
  - Το α καθορίζει την εναλλαγή των σελίδων και συνήθως είναι ίσο με 0.1

PageRank

- Ο χρήστης επισκέπτεται ένα κόμβο υ για ένα σταθερό χρόνο π(υ) που εξαρτάται:
  - Από τη δομή του γράφου
  - Την τιμή του α
- Καλούμε το π(υ) ως την τιμή PageRank του υ

Αλυσίδες Markov

- Μια αλυσίδα Markov είναι μια στοχαστική διεργασία διακριτού χρόνου (discrete-time stochastic process)
- Είναι μια διεργασία που συμβαίνει σε μια σειρά χρονικών βημάτων σε καθένα από τα οποία λαμβάνεται μια τυχαία επιλογή
- Μια αλυσίδα Markov απαρτίζεται από N καταστάσεις

Αλυσίδες Markov

- Οι αλυσίδες Markov χαρακτηρίζονται από ένα NXN πίνακα P
- Ο πίνακας P περιλαμβάνει τις πιθανότητες μετάβασης (transition probabilities)
- Κάθε κελί του πίνακα περιέχει τιμές στο διάστημα [0,1]
- Τα περιεχόμενα μιας γραμμής αθροίζουν το 1
- Σε κάθε χρονικό βήμα, μια αλυσίδα μπορεί να είναι σε μια μόνο κατάσταση από τις N διαθέσιμες
- Η καταχώρηση P_{ij} μας δείχνει την πιθανότητα να βρεθούμε στην κατάσταση j εφόσον είμαστε στην κατάσταση i
- Η καταχώρηση P_{ij} είναι γνωστή σαν πιθανότητα μετάβασης και εξαρτάται μόνο από την κατάσταση i – ιδιότητα Markov

Αλυσίδες Markov

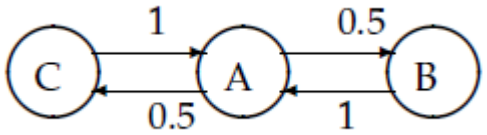
- Φορμαλισμός:

$$\forall i, j, P_{ij} \in [0, 1]$$
$$\forall i, \sum_{j=1}^{N} P_{ij} = 1$$

- Ο πίνακας με μη αρνητικά στοιχεία που ικανοποιεί την παραπάνω εξίσωση ονομάζεται στοχαστικός (stochastic matrix)

Αλυσίδες Markov

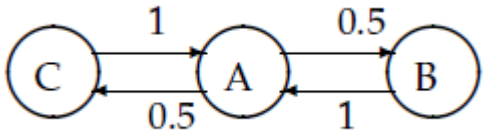
- Όπως είδαμε, η πιθανότητα μετάβασης εξαρτάται από την κατάσταση στην οποία βρισκόμαστε και όχι από το πως φτάσαμε σε αυτή την κατάσταση
- Αριστερά, βλέπουμε ένα παράδειγμα μιας αλυσίδας Markov με τρεις καταστάσεις
- Από την κατάσταση A, προχωρούμε στις B, C με ίδια πιθανότητα (0.5)
- Από τις B, C προχωρούμε στην A με πιθανότητα 1
- Ο πίνακας πιθανοτήτων μετάβασης φαίνεται αριστερά



$$\begin{pmatrix} 0 & 0.5 & 0.5 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Αλυσίδες Markov

- Οι πιθανότητες μετάβασης μπορούν να θεωρηθούν ως ένα διάνυσμα
- Προφανώς, τα περιεχόμενα του διανύσματος πρέπει να αθροίζουν στο 1
- Ένα N-διάστατο διάνυσμα κάθε στοιχείο του οποίου αντιστοιχεί σε μια κατάσταση μπορεί να θεωρηθεί ως η κατανομή πιθανότητας των διαθέσιμων καταστάσεων
- Για την αλυσίδα του παραδείγματος θα έχουμε τρία στοιχείο του 3-διάστατου διανύσματος



$$\begin{pmatrix} 0 & 0.5 & 0.5 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Αλυσίδες Markov

- Ένας χρήστης που πλοηγείται τυχαία στο Διαδίκτυο μπορεί να θεωρηθεί ως μια αλυσίδα Markov
- Κάθε κατάσταση είναι και μια σελίδα
- Η πιθανότητα μετάβασης σχετίζεται με την πλοήγηση σε άλλες σελίδες
- Η λειτουργικότητα teleport συνεισφέρει σε αυτές τις πιθανότητες

Αλυσίδες Markov

- Ο πίνακας γειτνίασης Α του γράφου θα οριστεί ως ακολούθως:
  - Αν υπάρχει σύνδεσμος από τη σελίδα i προς τη σελίδα j τότε θα έχουμε $A_{ij}=1$ διαφορετικά $A_{ij}=0$
- Στη συνέχεια μπορούμε να ορίσουμε τον πίνακα P των πιθανοτήτων μετάβασης του NXN πίνακα Α:
  - Αν μια γραμμή δεν έχει 1, τότε αντικαθιστούμε κάθε στοιχείο με 1/N – Για τι υπόλοιπες γραμμές κάνουμε το εξής:
  - Διαιρούμε κάθε 1 με το πλήθος των 1 σε κάθε γραμμή – αν έχουμε τρία 1, τότε θα τα αντικαταστήσουμε με 1/3
  - Πολλαπλασιάζουμε τον πίνακα που προκύπτει με 1-α
  - Προσθέτουμε α/N σε κάθε αποτέλεσμα για να πάρουμε την τελική μορφή του P

Αλυσίδες Markov

- Σε κάθε χρονικό βήμα μπορούμε να απεικονίσουμε την πιθανότητα της θέσης του χρήστη μέσω ενός διανύσματος πιθανοτήτων $\vec{x}$
- Σε χρόνο t=0, ο χρήστης μπορεί να ξεκινήσει σε μια κατάσταση για την οποία το $\vec{x}$ είναι 1 και όλες οι άλλες είναι 0
- Η κατανομή πιθανότητας σε χρόνο t=1 θα δοθεί από το διάνυσμα πιθανοτήτων $\vec{x}$P
- Σε χρόνο t=2 δίνεται από το διάνυσμα $(\overrightarrow{x}\mathrm{P})\mathrm{P}=\vec{x}\mathrm{P}^2$
- Γίνεται αντιληπτό πως με βάση την αρχική κατανομή των πιθανοτήτων του P μπορούμε στη συνέχεια να υπολογίσουμε τις πιθανότητες μετάβασης των χρηστών σε κάθε χρονικό σημείο

Αλυσίδες Markov

- Μια αλυσίδα Markov ονομάζεται ergodic εφόσον υπάρχει θετικός ακέραιος T τέτοιος ώστε για όλα τα ζεύγη καταστάσεων i,j αν εκκινήσουμε σε χρόνο 0 στην κατάσταση i τότε για κάθε t>T, η πιθανότητα να είμαστε στην κατάσταση j σε χρόνο t είναι μεγαλύτερη του 0
- Για να είναι μια αλυσίδα ergodic πρέπει να ισχύουν δύο συνθήκες:
  - Irreducibility: εξασφαλίζει ότι υπάρχει μια ακολουθία μεταβάσεων από μια κατάταση σε μια άλλη με μη μηδενική πιθανότητα
  - Aperiodicity: εξασφαλίζει ότι όλες οι καταστάσεις δεν τμηματοποιούνται σε υποσύνολα τέτοια ώστε όλες οι μεταβάσεις γίνονται κυκλικά από το ένα σύνολο στο άλλο

Αλυσίδες Markov

- Θεώρημα:
  - Για κάθε ergodic αλυσίδα, υάρχει ένα μοναδικό διάνυσμα πιθανοτήτων $\vec{\pi}$ το οποίο είναι το αριστερό ιδιοδιάνυσμα (eigenvector) του P τέτοιο ώστε αν η(i,t) είναι το πλήθος των επισκέψεων στην κατάσταση i σε χρόνο t τότε

$$\lim_{t\to\infty} \frac{\eta(i,t)}{t} = \pi(i)$$

Όπου π(i)>0 είναι η steady state πιθανότητα της κατάστασης i

Αλυσίδες Markov

- Από το προηγούμενο θεώρημα εξάγεται πως:
  - Ένας ‘τυχαίος’ περίπατος με teleporting θα έχει σαν αποτέλεσμα μια μοναδική κατανομή των steady state πιθανοτήτων πάνω από τις καταστάσεις της αλυσίδας Markov
  - Αυτή η πιθανότητα για μια κατάσταση θα είναι η τιμή PageRank της αντίστοιχης σελίδας στο Διαδίκτυο

Υπολογισμός PageRank

- Τα αριστερά ιδιοδιανύσματα του P είναι N-διανύσματα $\vec{\pi}$ τέτοια ώστε:
  $\vec{\pi}$P=λ$\vec{\pi}$
- Οι N καταχωρήσεις στο πρωταρχικό ιδιοδιάνυσμα $\vec{\pi}$ είναι οι πιθανότητες του τυχαίου περιπάτου με teleporting
- Αυτές οι τιμές είναι οι τιμές PageRank για τις αντίστοιχες σελίδες

Υπολογισμός PageRank

- Υπάρχουν αρκετοί αλγόριθμοι για τον υπολογισμό των ιδιοδιανυσμάτων
- Μια εισαγωγική μέθοδος είναι η power iteration
- Αν $\vec{x}$ είναι η αρχική κατανομή πιθανοτήτων των καταστάσεων τότε η κατανομή σε χρόνο t είναι $\vec{\pi}P^t$
- Όσο το t μεγαλώνει, αναμένουμε η κατανομή $\vec{\pi}P^{t2}$ να είναι όμοια με την $\vec{\pi}P^{t+1}$
- Η μέθοδος power iteration προσομοιώνει ένα τυχαίο περίπατο
- Εκκινεί από μια κατάσταση και εκτελεί τον περίπατο για ένα μεγάλο αριθμό βημάτων t
- Ταυτόχρονα καταγράφει τις συχνότητες επίσκεψης
- Μετά από ένα μεγάλο αριθμό επαναλήψεων οι συχνότητες συγκλίνουν προς την τελική τιμή
- Συνήθως τερματίζουμε όταν οι μεταβολές των συχνοτήτων είναι κάτω από ένα συγκεκριμένο όριο

Υπολογισμός PageRank

- Έστω γράφος με τρεις κόμβους 1,2,3. Οι σύνδεσμοι έχουν ως εξής: 1→2, 3→2, 2→1, 2→3. Επίσης, α=0.5
- Ο πίνακας μεταβάσεων έχει ως εξής:

$$P = \begin{pmatrix} 1/6 & 2/3 & 1/6 \\ 5/12 & 1/6 & 5/12 \\ 1/6 & 2/3 & 1/6 \end{pmatrix}$$

- Αν ο χρήστης εκκινήσει από τη σελίδα 1 που αντιστοιχεί στο διάνυσμα $\vec{x_0} = (1\ 0\ 0)$ τότε μετά από ένα βήμα θα έχουμε:

$$\vec{x_0}P = \begin{pmatrix} 1/6 & 2/3 & 1/6 \end{pmatrix} = \vec{x_1}$$

- Μετά από δύο βήματα:

$$\vec{x_1}P = \begin{pmatrix} 1/6 & 2/3 & 1/6 \end{pmatrix} \begin{pmatrix} 1/6 & 2/3 & 1/6 \\ 5/12 & 1/6 & 5/12 \\ 1/6 & 2/3 & 1/6 \end{pmatrix} = \begin{pmatrix} 1/3 & 1/3 & 1/3 \end{pmatrix} = \vec{x_2}$$

Υπολογισμός PageRank

- Μετά από πολλαπλά βήμα:

$\vec{x}_0$	1	0	0
$\vec{x}_1$	1/6	2/3	1/6
$\vec{x}_2$	1/3	1/3	1/3
$\vec{x}_3$	1/4	1/2	1/4
$\vec{x}_4$	7/24	5/12	7/24
...	...	...	...
$\vec{x}$	5/18	4/9	5/18

- Βλέπουμε πως συγκλίνουμε στο διάνυσμα:

$$\vec{x} = (5/18 \quad 4/9 \quad 5/18)$$

Υπολογισμός PageRank

- Οι τιμές PageRank είναι ανεξάρτητες από τα ερωτήματα των χρηστών
- Οι τιμές αναπαριστούν την ποιότητα των σελίδων
- Όμως, πολλές φορές η κατάταξη των σελίδων θα πρέπει να διαφοροποιεί΄ται ανάλογα με το ερώτημα
- Για αυτό το λόγο οι μηχανές αναζήτησης υιοθετούν τις τιμές PageRank μαζί με άλλες μετρικές πριν προσφέρουν το τελικό αποτέλεσμα