Ανάκτηση πληροφορίας

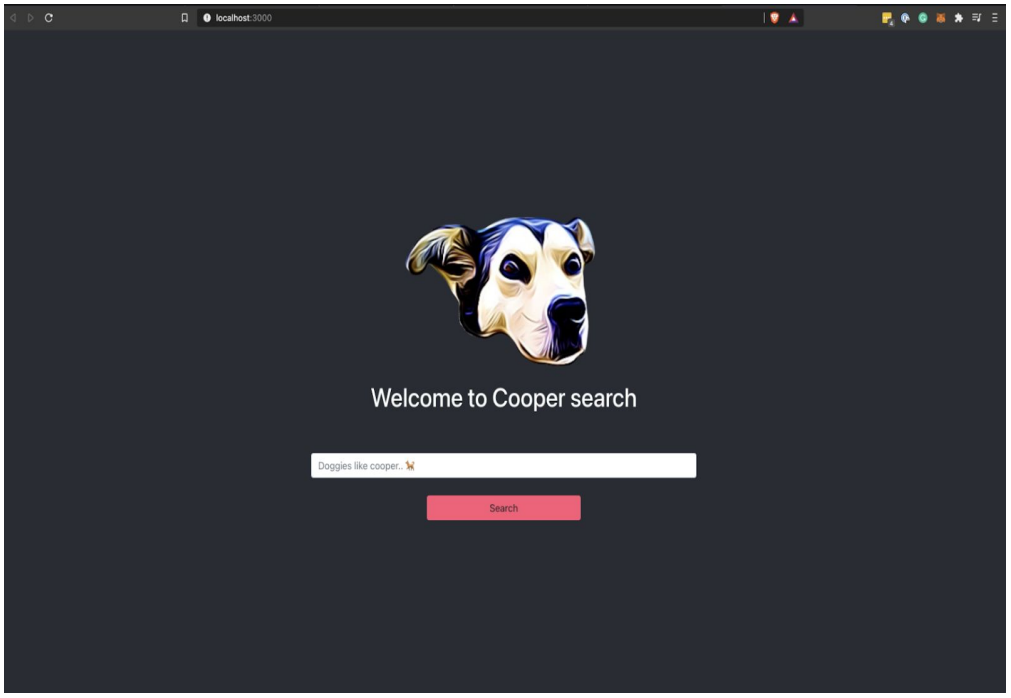
Βασίλης Πιτσιάβας (2859)
Δημήτρης Λαμπρινός (2761)

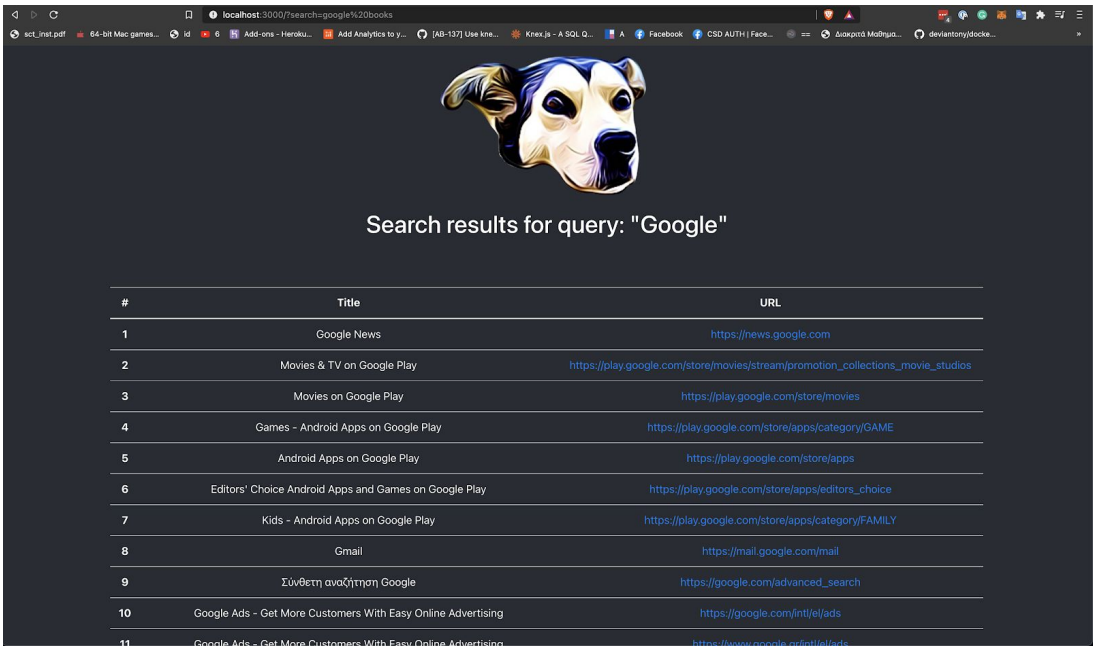
Αυτή η εργασία έχει ως σκοπό τη δημιουργία μιας ολοκληρωμένης μηχανής αναζήτησης. Στην υλοποιηση μας περιέχεται ένα web interface γραμμένο σε javascript (React.JS) που περιλαμβάνει τόσο την βασική ιστοσελίδα που περιέχει την φόρμα της ερώτησης και τον πίνακα των αποτελεσμάτων, αλλά και το component που επικοινωνεί με τον server. Επίσης περιλαμβάνεται ένα backend σε GoLang που περιέχει όλη την λογική του tf-idf, ένα cli tool που πυροδοτεί τον crawler και αποθηκεύει τα δεδομένα σε μια sqlite βάση καθώς και ένα API service που σερβίρει τα αποτελέσματα στο react frontend.

Components

Web interface

Το component αυτό περίεχει όλο το interface που θα χρησιμοποιούν οι end users για να χρησιμοποιήσουν την μηχανή αναζήτησης. Γραμμένο σε react.js/css, περιέχει 2 κύριες σελίδες. Αυτή της βασικής σελίδας που περιέχει την φόρμα της ερώτησης, καθώς και την σελίδα που περιέχει τα αποτελέσματα. Για την επικοινωνία με το backend γίνονται requests με JSON payloads. Το interface μοιάζει ως εξής:





Crawler

Ο crawler αποτελεί το βασικό backend της μηχανής αναζήτησης. Χρησιμοποιώντας τον γνωστό αλγόριθμο BFS (σε mutlithreaded μορφή) προσπελαύνει κάθε σελίδα ξεκινώντας απο ένα base site δίνοντας τα δεδομένα στον indexer για να κάνει indexing των ιστοσελίδων. Στην υλοποίηση μας ο crawler είναι σε θέση να θεωρεί διαφορετικές τις σελίδες που βρίσκονται είτε σε διαφορετικό path είτε στο ίδιο path αλλά περιέχουν διαφορετικά uri parameters. Επίσης ο crawler παίρνει ως παραμέτρους το πλήθος των threads που θέλουμε να χρησιμοποιήσει, το όριο των ιστοσελίδων που θέλουμε να επισκεφτει καθώς και αν πρέπει να επισκεφτεί ξανά τις σελίδες που έχει ήδη επισκεφτεί. Για το persistence των δεδομένων χρησιμοποιήσαμε μια sqlite βάση που περιέχει τα term frequencies της κάθε ιστοσελίδας καθως και το global document frequency του search engine.

Indexer

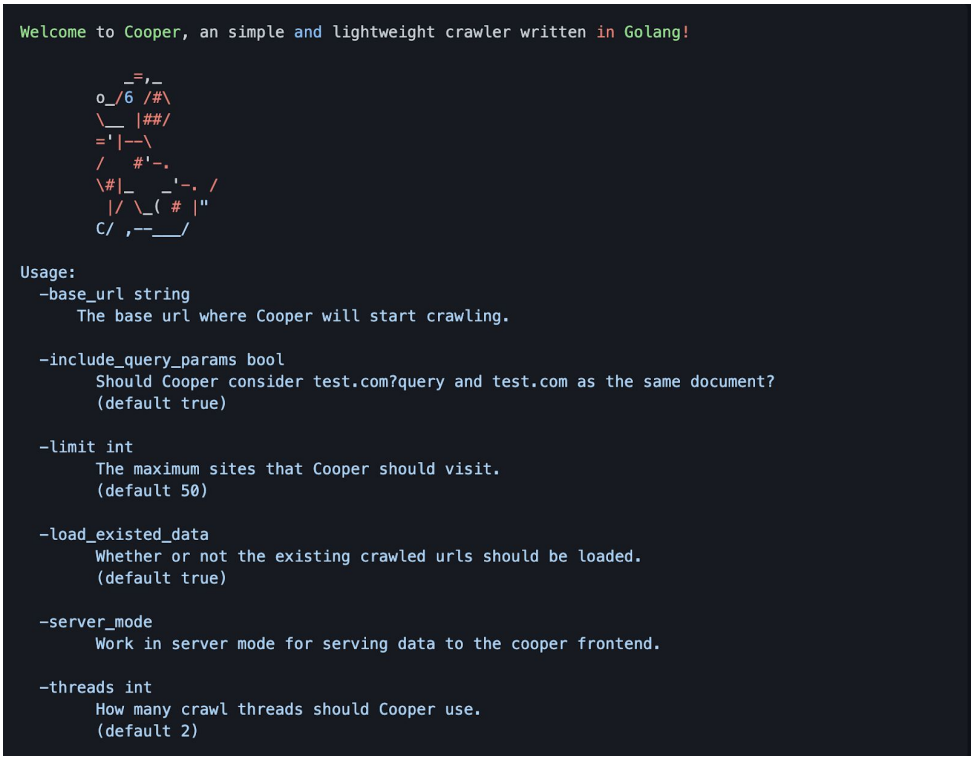
Ο indexer υλοποιει την τεχνική tf-idf για την εύρεση των πιο σημαντικών terms ενός collection λέξεων και κάνει update τον document frequency πίνακα. Επίσης επειδή προέκυψε πολλές φορές το πρόβλημα της θεώρησης παραγωγων λέξεων ως διαφορετικες χρησιμοποιήσαμε ένα stemming library που είναι γραμμένο σε go για να βρίσκουμε το stem της κάθε λέξης. Επειδή ο indexer περιέχει όλη τη λογική του search engine γράψαμε tests για αυτο το component που μπορούν να βρεθούν στον κώδικα.

API

Για την επικοινωνία του frontend με το backend δημιουργήσαμε ένα mini RESTful JSON API σε GoLang. Το API αυτό περιέχει ένα GET endpoint που δέχεται την ερώτηση του χρήστη και απαντάει μια λίστα με τις πιο relevant απαντήσεις.

CLI

Για την εκίνηση του crawler ή του server δημιουργηθηκε ένα cli tool σε GoLang που με τις καταλληλες παραμέτρους τρέχει τον crawler η ανοίγει τον server ωστε να μπορεί να τρέξει το frontend. Το cli μοιάζει ως εξής:



Χρήση του Cooper

- Για την χρηση του cooper αρχικά πρέπει να τρέξει ο crawler ετσι ώστε να γίνουν index ιστοσελίδες. Για να γίνει αυτό τρέχουμε:
  - cd crawler για να πάμε στο package του crawler
  - ./cooper -base_url=https://google.com για να τρέξει ο crawler
  - ./cooper -server_mode για να τρέξει ο server (default :8080)

- Στη συνέχεια για να τρέξει το frontent χρειάζεται να ανοίξουμε τον react server για να γίνει serve το web interface.

- cd ../frontend για να πάμε στο react service
- yarn start για να τρέξει το react και να ανοιξει το web search.

Η εργασία είναι επίσης ανεβασμένη στο github με url:
https://github.com/pkakelas/cooper