Sanaindeksi
Toteutusdokumentti

Patrik Ahvenainen*
Helsingin yliopisto
Fysiikan laitos

1. syyskuuta 2012

1 Yleisrakenne

Tämän ohjelman koodi koostuu kolmesta paketista: Trees, Support ja WordIndex. Näitä käsitellään enemmän seuraavissa. Pakettien, moduulien, luokkien ja metodien kuvaukset löytyvät yksityiskohtaisemmin doc-hakemistosta joko PDF-muodossa (`API documentation.pdf`) tai HTML-muodossa (päähakemisto tiedostossa `index.html`.

1.1 Trees

Tämä paketti sisältää työn toteutukseen käytetyt puurakenteet.

Ohjelman tietorakenteina käyetään Trie- ja punamustaa puuta. Kummatkin puut toteuttavat Tree-rajapinnan. Lisäksi Trie toteuttaa PartialTree-rajapinnan, joka mahdollistaa myös sanojen alkujen etsimisen. Rajapinnat on toteutettu Pythonissa ABCMeta-luokan [viite] avulla. Puut ovat vastuussa indeksoinnista ja sanan (tai sen alun) löytämisestä indeksistä.

1.2 Support

Support-paketti sisältää kaksi ohjelman tueksi kirjoitettua moduulia. Tiedosto-operaatiota varten on DataHandling-moduuli ja sanojen esiintymien tallentamista varten on toteutettu yksinkertainen LinkedList-luokka, joka on kahteen suuntaan linkitetty lista.

1.3 WordIndex

Searcher-luokka taas on vastuussa hakujen suorittamisesta käyttäen Puu-rajapintaa. Sanojen lukemiseen tiedostosta käytetään WordReader-luokkaa,

*patrik.ahvenainen@alumni.helsinki.fi / 013326292

joka vastaa myös sanojen muokkaamisesta hyväksyttävään muotoon (lähinnä Trien indeksointia varten).

Tiedostojen indeksointia voi testata pysanaindeksi-skriptillä, jonka avulla voi testata kumpaakin puuta sekä tehdä hakuja Materials-kansiosta löytyviin tiedostoihin. Kyseessä on siis pääohjelma työssä toteutetuille luokille. Testaukseen liittyvät tiedostot on rapotoitu testausokumentissa. Ohjelman toiminnan voi nopeasti testata pyUnitTest-moduulilla.

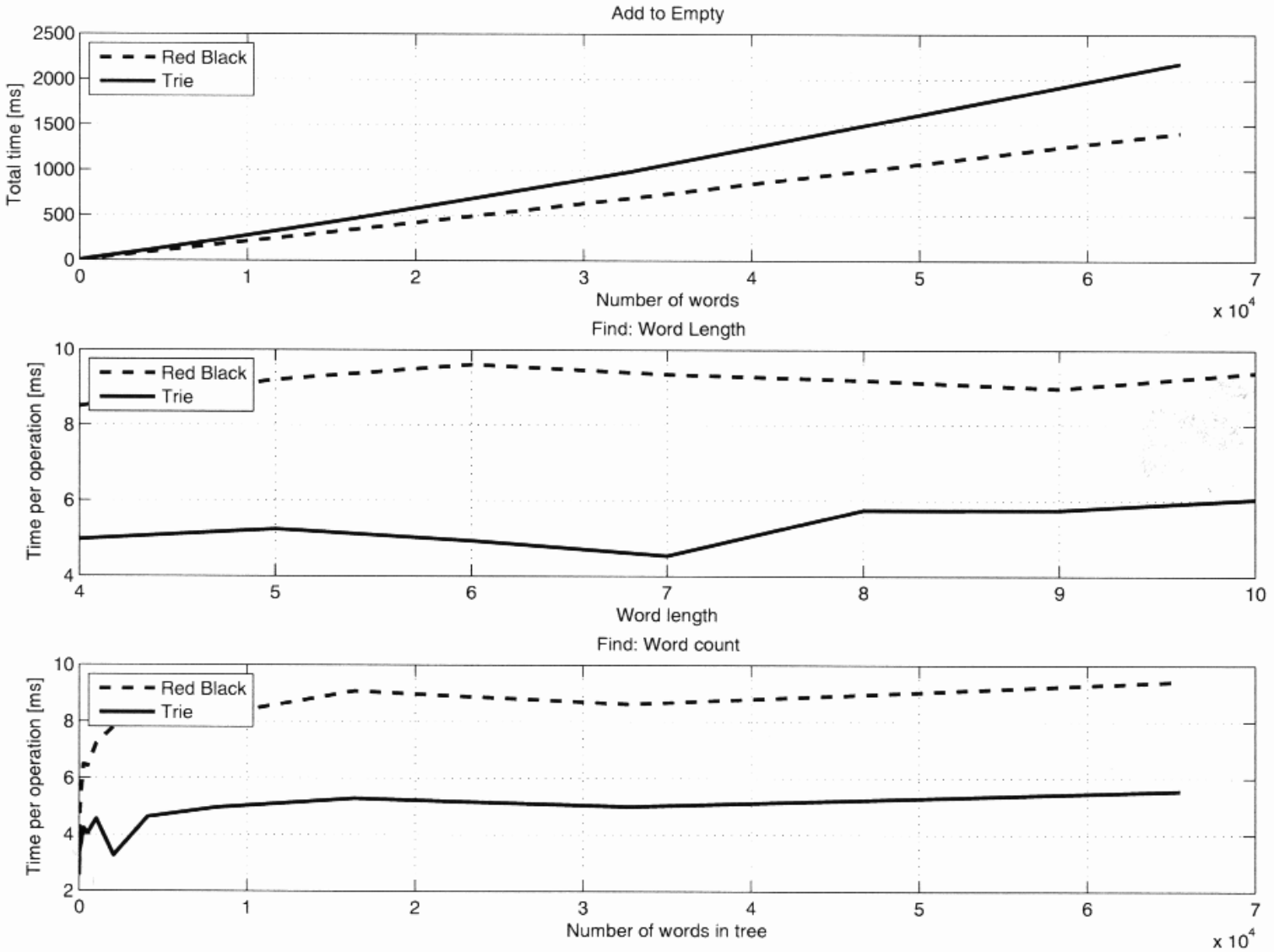
2 Aika- ja tilavaativuudet

Aikatestit on esitetty seuraavalla sivulla. Testejä varten kirjoitettiin timing-moduuli. Kuvaajat aikavaativuuksista piirrettiin Matlab-skriptillä.

Ensimmäisessä testissä testattiin sanojen lisäämisaikaa. Punamusta puu oli huomattavasti nopeampi. Tämä johtunee siitä, että Trie on prefix-puu, eli sillä voidaan etsiä myös sanojen alkua, toisin kuin RedBlack-puu.

Toisessa testissä kokeiltiin sanojen lisäämisen aikavaativuutta sanan pituuden funktiona. Sanojen pituuden testaamisen haasteena on se, että englanninkielisiä sanoja on runsaasti vain noin 4-10 merkin pituisena. Tätä lyhyemmät tai pidemmät sanat ovat harvinaisia (tai poikkeuksellisia), joten niillä ei voi testata haun nopeutta. Trie-puu näyttää kuitenkin noudattavan suunnilleen odotettua $\mathcal{O}(l)$-aikavaativuutta, jossa l on sanan pituus. Toisaalta punamusta puu ei näyttänyt välittävän oleellisesti sanan pituudesta. Kaikkien sanojen etsimisessä Trie-puu oli nopeampi (alle 6 ms per sana, kun punamustan puun keskimääräinen hakuaika oli aina vähintään 8 ms sanaa kohti).

Kolmannessa testissä testattiin sanojen hakemisen nopeutta puun sisältämien sanojen lukumäärän funktiona. Kummatkin puut olivat huomattavasti nopeampia pienillä puilla kuin isommilla. Isommilla puilla Trie näytti kuitenkin vakiintuvat lähestulkoon vakiotasolle, eli riittävän suurilla puilla aikavaativuus ei riippunut sanojen määrästä. Punamusta puu taas näytti hiukan hidastuvan isommilla puilla. Tämä on loogista, sillä teoreettinen aikavaativuus puun koosta (sanojen määrä n) hakuoperaatiolle punamustasta puusta on $\mathcal{O} = \log(n)$.


Add to Empty
Red Black
Trie
Total time [ms]
2500
2000
1500
1000
500
0
0
1
2
3
4
5
6
7
Number of words
x 10^4
Find: Word Length
Red Black
Trie
Time per operation [ms]
10
8
6
4
4
5
6
7
8
9
10
Word length
Find: Word count
Red Black
Trie
Time per operation [ms]
10
8
6
4
2
0
1
2
3
4
5
6
7
Number of words in tree
x 10^4

3 Suorituskyky

Suorituskykyä vertailtaessa haasteeksi tuli oikean aineiston käyttö. Eli kun indeksoitavana oli oikea aineisto täytyi siitä etsiä myös oikeita sanoja. Tämä siksi, että etsiminen lopetetaan kun seuraavaa solmua ei enää löydy. Sanojen täytyy siis edustaa samanlaista kirjainjakaumaa kuin indeksoitavan tekstin. Tässä kyseinen ongelma ratkaistiin muodostamalla lista n:n pituisista satunnaisista englanninkielen sanoista. Sanojen pituudeksi rajattiin 4-10 merkkiä, koska muun pituisia sanoja on vähemmän. Satunnaissanalista muodostettiin ottamalla Internet-palvelimelta [1] satunnaisia sanoja.

Sanojen satunnaisuuden lisäksi keskiarvoistettiiin kaikki mittaustulokset useamman (peräkkäin suoritetun) toiston yli.

4 Parannusehdotuksia

Nyt toteutus toimii vain UTF-8 –tyyppisiin tiedostoihin. Muut fonttikoodaukset voitaisiin luettaessa muuntaa tämän tyyppisiksi, jolloin luettava tiedosto voisi olla minkä tyyppinen tahansa.

Aikavertailun kannalta olisi kätevää, jos Trien indeksoinnin voisi toteuttaa myös punamustaan puuhun verrattavalla tavalla, eli ilman sanojen alkujen indeksointia.

Searcher-luokka voisi vaatia, että hakija toteuttaa Finder-rajapinnan, jossa on määritelty etsimismetodin kutsu. Sinäänsä Searcherin ei nimittäin tarvitsisi vaatia hakijan olevan juuri hakupuu, kunhan hakumetodi löytyy.

Lähteet

[1] Random Word API. http://randomword.setgetgo.com/. Haettu 1.9.2012.