УНИВЕРЗИТЕТ „ГОЦЕ ДЕЛЧЕВ“ – ШТИП

ФАКУЛТЕТ ЗА ИНФОРМАТИКА

Информациски технологии

Штип

ЃОРЃЕ ГИЧЕВ

НАПРЕДНО ПРЕБАРУВАЊЕ ИНФОРМАЦИИ КАЈ ERP АПЛИКАЦИИ

- МАГИСТЕРСКИ ТРУД -

Штип, Јули 2014

КОМИСИЈА ЗА ОЦЕНКА И ОДБРАНА НА МАГИСТЕРСКИ ТРУД

Претседател: **Доц. д-р Зоран Здравев**
Професор на факултетот за информатика
Универзитет ,,Гоце Делчев" – Штип

Член (ментор): **Проф. д-р Јован Пехчевски**
Професор на факултетот за информатика
Европски универзитет Р.Македонија

Член: **Доц. д-р Игор Стојановиќ**
Професор на факултетот за информатика
Универзитет ,,Гоце Делчев" – Штип

Датум на одбрана : 11.07.2014год.

Ѓорѓе Гичев

Напредно пребарување информации кај ERP апликации

Универзитет „Гоце Делчев" – Штип

Декларација

Потврдувам дека, освен за случаи каде е направено прописно цитирање, оваа магистерска теза е оригинално дело на самиот автор; тезата не е поднесена претходно, во целост или делумно, да се квалификува за каква било друга академска награда; содржината на тезата е резултат на работата која е извршена од официјалниот датум на одобрување на магистерската пријава; и, секоја уредувачка работа, платена или неплатена, извршена од страна на трети лица е прописно референцирана.

Со цел делумно исполнување на барањата за степенот Магистер по Информатика, потврдувам дека ги имам објавено следните научно-истражувачки трудови:

1 Гичев, Ѓ, Атанасова, И, Пехчевски, Ј, (2012). *Подобрување на безбедноста на Cloud-базираните ERP системи*. Зборник на Факултетот за информатика при Универзитетот „Гоце Делчев“ – Штип, (1)1: 77-87.

2 Гичев, Ѓ, Пандевска, А, Атанасова, И, Здравев, З, Мартиновска, Ц, Пехчевски, Ј, (2014). *Пребарување информации во ЕРП системи: АртАИИС студија на случај*. Зборник на Факултетот за информатика при Универзитетот „Гоце Делчев“ – Штип.

Магистерската теза е напишана во Word 2013 на Windows 8.1 платформа.

_____

Ѓорѓе Гичев
Факултет за Информатика
Универзитет Гоце Делчев – ШТИП
Скопје, Јули 2014

СОДРЖИНА:

Листа на Слики

Листа наТабели

Листа на Изворни кодови

Апстракт

Од голема важност во процесот на раководење на една компанија е богатството информации со кои располага компанијата да биде постојано достапно на оној на кого му е потребно. Ваквите информации се употребуваат за раководењето со работните процеси во реално време и секојдневното анализирање на податоците потребни за раководење, комуникација и контрола на процесите на компанијата. ЕРП системите, кои денес го претставуваат јадрото на информации на бизнисот на една компанија, се тема во истражувачките кругови поради постојаните потреби за нивно подобрување и редефинирање на функционалните аспекти на деловните системи.

Во овој магистерски труд е прикажана практична имплементација на пребарувачка машина за постоечки ЕРП систем АртАИИС. Претставено е текст базираното пребарување како можност за побрзо и поедноставно пребарување во ЕРП системот АртАИИС, кое овозможува искористување на богатството на информации во компанискиот ИС, како за постојани, така и за ад хок и необучени корисници за увид, бизнис аналитика и сл. Евалуацијата на стандардното филтер пребарување и пребарувачката машина за дадена колекција за тестирање на информациски потреби покажува дека пребарувањето базирано на текст е полесно за користење низ ЕРП системот АртАИИС за сите типови на корисници.

1 Вовед

Информацијата било каде во било кое време - е неопходност што на човекот во 21от век му се наметнува како потреба во неговото секојдневие и работен живот и како таква го мотивира постојано да ги подобрува и оптимизира методите и алатките за пребарување. Во процесот на раководење на една компанија е круцијално информациите со кои располага компанијата,кои укажуваат на нејзиното функционирање и развој, кои се употребуваат за подобрување на целосниот тек на работа, и оние кои се потребни за секојдневно работење, да бидат постојано и брзо достапни на оној на кого му се потребни. ЕРП системите кои денес го претставуваат јадрото на информации на бизнисот на една компанија, сè повеќе наоѓаат место во истражувачките кругови поради потребите за нивно подобрување и за збогатување на функционалностите на системите. Од посебен интерес во последно време е потребата за брзо и ефикасно пребарување низ содржините на ЕРП системот без притоа да се користат сложените пребарувачки механизми кои се историски составен дел на ЕРП системите. Пребарувањата ‘google like’ станаа defacto стандард и корисниците на ЕРП системите бараат да имаат такви можности и за пребарување низ нивните бази на податоци.

1.1 Цели на истражувањето

Целта на оваа магистерска тема е дизајнирање на пребарувачка машина за т.н. физички непостоечки документи (фактури, требовници, испратници и сл.) во рамки на еден ЕРП систем (во ЕРП системите документите се чуваат низ разни табели во базата на податоци, а не како целосни документи). Пребарувањето треба да резултира во приказ на релевантните документи во моментот на пребарување во соодветен формат (pdf, word, и сл.) достапен за преглед, печатење и евиденција. Во магистерската тема предизвик е дизајнирањето на пребарувачка машина која поддржува пред сè динамичко индексирање (променлива колекција на документи) и која со дефинирање на неколку параметри се прилагодува за пребарување на било кој тип на документ потребен на корисникот на ЕРП системот.

Пребарувачката машина ќе биде споредена со веќе постоечки решенија на пребарување информации, но целта е нејзината ефикасност и ефективност да се погледнат во полето на пребарување во реално време на ЕРП ‘документи’ кое како такво не постои во моментот.

1.2 Истражувачки прашања

Проблематиката се состои од четири поставени прашања на кои ќе биде соодветно даден одговор во текот на истражувањето:

1. Која е потребата од воведување техники за пребарување информации кај ЕРП системите користени од страна на компаниите?
2. Кои компоненти ќе ги содржи една таква пребарувачка машина и на кој начин истата би можела да се интегрира со постоечки ЕРП систем?
3. Како може да се мери ефективноста и ефикасноста на пребарувачката машина од кориснички аспект?
4. Која е користа за компаниите од примена на машина за пребарување информации низ ЕРП систем?

1.3 Ограничувања

Во магистерскиот труд се поставени следните ограничувања:

- Пребарувачката машина е имплементирана и евалуирана само над еден ЕРП систем, АртАИИС.
- Пребарувачката машина е развиена за пребарување на виртуелни/електронски документи, но не и за други типови на документи.
- Споредбата на пребарувачката машина и стандардното пребарување во ЕРП системот е изведена над колекција од документи од тип Тикети, но не и од дополнителни категории на документи. Причината поради која не се користеше јавно достапна тест документ колекција е недостигот и непостоењето на ваков тип на тест колекција која покрива ЕРП и останати компаниски ИС.

1.4 Придонеси на магистерскиот труд

Пребарувањето информации низ компаниски информациски систем е задолжителна фукционална категорија во денешниот свет на милиони податоци кои се генерираат во овие системи. Денешните компаниски ИС најчесто не нудат опција за пребарување и добивање на информации со која би се опфатил сите релевантни подсистеми кои потенцијално ги содржат ваквите информации, и тоа истовремено да биде брзо и едноставно за било каков корисник на системот. Повеќето големи ЕРП вендори, свесни за потребите на корисниците од пребарувачките машини, во последните 2 години вложуваат голем напор и воведуваат надградби на своите постоечки ЕРП решенија кои

би го скратиле значително времето на добивање на релевантни информации.
Овој магистерски труд придонесува во воведување на пребарување низ ЕРП системот АртАИИС не само за обучени и редовни корисници за дадени функционалности туку и за корисник кој е нов или нема познавање од одреден дел на информациониот систем. Пребарувачката машина креирана и претставена во овој труд е дизајнирана и развиена да може да се интегрира и биде дел од било кој ИС преку концептот на виртуелни документи. Трудот се фокусира пред сè на навиките на корисник, кој со едноставно текст пребарување слично на Google пребарувачот, доаѓа брзо и едноставно до бараните податоци. Магистерскиот труд ги оценува и споредува двата типа на пребарување над податоците во АртАИИС системот.

1.5 Преглед на магистерскиот труд

Магистерскиот труд е организиран во неколку поглавја. Во второто поглавје се разгледани главните карактеристики на веб пребарувањето и IR системите. Разгледани се моменталните трендови и имплементации како и архитектурата на ваквиот тип на системи. Во ова поглавје е претставено компаниското пребарување како специјален тип на пребарување низ компаниските ИС над структурирани и неструктурирани податоци. Детално е разгледана архитектурата и структурата на една пребарувачка машина со сите нејзини основни компоненти и фази.
Во третото поглавје е даден вовед и опис на главните функционалности и карактеристики на Enterprise Resource Planning (ERP) системите. Разгледана е нивната еволуција, и дополнително карактеристиките за начините на пребарување информации.
Четвртото поглавје ги претставува постоечките методологии за оценување на IR системите. Прикажани се стандардни тест колекции и методи на оценување како на нерангирани така и на рангирани резултати.
Во петтото поглавје е даден детален опис на практичната имплементација на Артисофт Пребарувачка Машина (во понатамошниот текст АПМ). Најпрво е даден увид во архитектурата на пребарувачката машина со вовед во технологијата за развој, интерниот и екстерниот дизајн и нејзината имплементација во АртАИИС ЕРП.
Во шестото поглавје е прикажан процесот на оценување и тестирање на претходно развиениот модел, како и експерименталните резултати за избраната тест колекција.
На крај магистерскиот труд прави осврт кон заклучните согледувања од резултатите и анализите од процесот на евалуација на двата начина на пребарување во ЕРП системот, заедно со препораки за можни идни истражувачки насоки и надградби на АПМ-от.

2 Веб пребарување и IR системи

Поради енормното количество податоци кои ги има Вебот, можноста да се пронајдат информации со употреба на алатки за пребарување како Google е есенцијално. Таквите алатки всушност го донесоа вториот бран на интернетот како светска информациска мрежа. Слични ефекти може да се добијат и при пребарување на информации во рамки на информатичките системи на една компанија. Компаниските апликации слично како и Интернетот, содржат податоци на компанијата, некогаш обединети во Интегрирани системи, а некогаш и како острови на информации расфрлени по серверите на компанијата. Како и во раните денови на Интернетот, за да се пронајде информација во компаниските деловни апликации потребно е да се знае каде истата се наоѓа. Овој метод на пребарување е вообичаен за редовни корисници на системот во разните служби на компанијата кои се занимаваат со набавка, продажба, производство, и слично. Но проблемите настануваат кога до информација сака да дојде некој што не е толку искусен во користењето на системот, или кој не знае каде треба да бара или пак и за искусните корисници кога треба да дојдат до некоја информација која не е од скорашен датум или не е баш од нивното поле на експертиза. Понекогаш за да се дојде до одредена информација треба да се пребараат повеќе разни апликации од Информациониот систем, а понекогаш и не се знае каде сè може да се најде одреден податок. Токму ова се мотивите да во рамки на компаниските информациони системи се имплементира пребарувачка машина која ќе ги обедини индиректно податоците за корисникот кој ги бара.

Пребарување низ компаниските информации е функционалност која, во своите деловни системи, ја имплементираат одредени технолошки компании, вклучувајќи го и Google (Google Search Appliance и Google Mini Products). SAP и Oracle имаат пребарувачки алатки како засебни продукти не само за своите апликации, туку и за било кои апликации и податоци во рамки на компаниите. Други компании, вклучувајќи ги Thunderstone, Index Engines, Autonomy, Convera, FAST Search и Verity, развиваат генерички пребарувачки уреди за пребарувања низ компаниските податоци. Пребарувањето во рамки на компанискиот информациски систем ја зголемува ефективноста на пребарувањето едно ниво над Google и сличните пребарувачки алатки, бидејќи нуди контекст во дизајнот на пребарувањето на корисникот (пребарувањето во рамки на компанискиот информационен систем знае што пребарува,

односно има контекст)

Преку интеграција на технологиите за пребарување директно во компаниските информациони системи, нивните дизајнери им овозможуваат на корисниците да специфицираат повеќе типови на информации во пребарувањето. Истото овозможува филтрирање на ирелевантните резултати и поефикасно добивање на крајни резултати за корисникот.

Во контекст на досега кажаното се наметнува потреба денешните компаниски апликации да интегрираат во себе пребарувачки функционалности како составни компоненти. На тој нашин пребарувањето во рамки на компаниска апликација ќе овозможи лесно доаѓање до информација преку унифициран интерфејс, и дополнително, бидејќи се работи за специфична компонента изградена за таа апликација, ќе нуди дополнителни предности над генералните пребарувачки продукти (како на пример на Oracle, Google, Thunderstone, Index Engines итн).

Освен погорекажаното дополнителни предности на ваквите интегрирани решенија вклучуваат:

- *Намалени трошоци.* Бидејќи алатката за пребарување е интегрирана како дел од самата апликација не повлекува дополнителни лиценцни трошоци или дополнителен хардвер. Дополнително, не постои потреба и трошок за интеграција на алатката за пребарување со апликацијата.
- *Безбедност.* Иако и засебно решение за пребарување има вградено добри безбедносни и сигурносни механизми, истите мора да бидат интегрирани и со постоечките привилегии дефинирани во апликациите кои се извори на податоци. На пример, информациите од финансово е потребно да бидат овозможени за приказ само до вработените кои имаат право да читаат такви информации.
- *Контекст.* Може да се користат контекстуални информации за да се добијат пофокусирани резултати. На сличен начин како што постоечките пребарувачки машини како Google овозможуваат со помош на неколку филтри и дополнителни алатки да се добие понасочено пребарување. Овој концепт може исто така да биде применет и во контекст на бизнис процесите, така што доколку даден корисник е вклучен во функции поврзани со финансии, тогаш тој вид на резултати да бидат нагласени при рангирањето и пресметката на релевантноста.

- *Намера.* Бидејќи алатка за пребарување во рамки на компанискиот Информационен Систем (ИС) има познавање и од метаподатоците на апликацијата, можно е на корисниците да им се обезбеди едноставен и лесно разбирлив начин за изразување на одредена намера со општа бизнис терминологија како на пр. “податоци за нарачки“, “податоци за продукт“ и слични опишувачи.
- *Хибридно пребарување.* Комбинирано пребарување низ разните информациони системи но и низ податочни структури кои не се дел од информационите системи, а кои постојат во една команија (разни споредни евиденции неинтегрирани од разни причини со главните информациони системи на компанијата).

2.1 Архитектура и карактеристики на IR системите

Во денешно време, стотици милиони луѓе секојдневно се вклучени во процес на пребарување на информации со употреба на постоечките веб пребарувачки машини. Information retrieval полека станува најдоминантниот начин за пристап до информации. Значењето на зборот information retrieval може да биде многу опширно. Според Manning et al. (2009) information retrieval може да се дефинира како:

“*Information retrieval (IR) претставува пронаоѓање материјал (вообичаено документи) од неструктурирана природа кој задоволува одредено информациско барање од голема колекција (вообичаено чувана на компјутери) “*

IR системите исто така се разликуваат и според обемот на информации со кои работат, и може да се наведат три значајни поделби:

1. *Веб пребарување* – кај веб пребарувањето системот мора да обезбеди пребарување над милијарди документи чувани на милиони компјутери. Веб пребарувањето е потребно да задоволи одредени предуслови меѓу кои собирање на документи за индексирање и ефикасно функционирање со голем обем на податоци.
2. *Лично пребарување* – Во последните години, оперативните системи во себе започнаа да интегрираат и информациско пребарување, како на пример Spotlight на Mac OS X на Apple или Instant Search на Windows Vista. Понатаму, повеќето

програми за електронска пошта најчесто не само што обезбедуваат пребарување туку и класификација на текстови за филтрирање на Junk и Spam е-маил пораки. Во овие случаеви предизвик е справувањето со широк спектар на типови на документи на еден персонален компјутер. Дополнително, оваа функционалност треба да биде едноставна за употреба и рационално да ги користи ресурсите на персоналниот компјутер.

3. *Компаниско и домен - специфично пребарување* – Пребарувањето во оваа категорија враќа резултати од колекции на документи и податоци како на пример: (а) внатрешни компаниски документи, (б) база на податоци на пациенти, (в) пребарувачки статии за биохемија и сл. Во овој случај документите се најчесто чувани на централизирани податочни системи на една или повеќе дедицирани машини.

2.2 Компаниско пребарување

Актуелноста и интересот за пребарувачките машини забележува пораст со развивањето на свесноста на организациите за можностите кои ги нуди пребарувањето во рамки на сопствените информациски системи. Пребарувањето во рамки на компанијата може да се интерпретира како пребарување на дигитални текстуални материјали кои ги поседува самата организација вклучително:

- Интерните Информациони Системи (ЕРП, ЦРМ, ХРМ, ДМС, ...)
- Веб страната
- Компанискиот интранет
- Било кое друго мултимедијално множество на електронски документи кои посоти во интерните датотеки.

Многу карактеристики на пребарувањето во рамки на компанијата претставуваат предизвици за дизајнерите на IR системите. Информацијата во една компанија може да биде структурирана или неструктурирана. Документите се креираат од различни извори додека метаподатоците може да бидат креирани според повеќе различни шеми, или воопшто да не постојат. Дополнително сите корисници имаат некакви привилегии за пристап до одредени податоци поради различните нивоа на пристап до податоците, па пребарувачките машини треба да ги земат сите овие параметри во предвид при обработка и презентација на податоците кои се предмет на пребарување.

Постојат различни согледувања и анализи дека вработените трошат значително време при пребарување на потребни информации (Baeza-Yates and Ribeiro-Neto, 2010).

Според IDC вработените поминуваат 9-10 часа неделно во пребарување на одредени информации кои им се потребни во секојдневното работење. Според Butler Group дури 10% од компаниските трошоци во платени часови се поради изгубеното време за пребарување на информации. Истражување на Accenture од 2007 покажува дека менаџерите поминуваат околу 2 часа секој ден пребарувајќи информации.

Во продолжение се наведени некои можни употреби и сценарија за користење на напредни техники на пребарување низ компаниски информациони системи. Еден типичен пример на користење е за добивање: (а) информации за трендови; (б) информации за пикови во продажба; (в) информации за типични барања на клиентите; (г) разни аналитички извештаи.

Понатаму, пребарувачки алатки во рамки на една компанија може да се користат за лоцирање на соодветна експертиза при составување на проектни тимови. Дополнително, автоматизирани интерни извештаи може да инкорпорираат информации од пребарувачките резултати. Во продолжение се дадени и неколку дополнителни примери на сценарија за употреба на напредно пребарување во компаниски информациски систем (Baeza-Yates and Ribeiro-Neto, 2010):

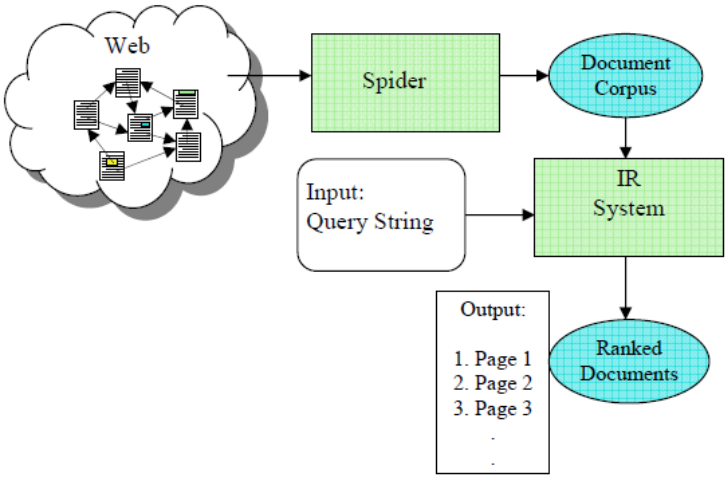
1. *Одобрување на барање за патување* - со цел одговорното лице во компанијата да може да одлучи дали да одобри барање за патување за вработен потребни се неколку информации кои треба да се достапни: колку години е вработениот во компанијата, дали компанијата ќе трпи загуби поради ова отсуство, каква е политиката на компанијата за вакви барања, колку има потрошено вработениот изминатата година на патувања, и сл. Ново вработен менаџер ќе има потреба да пребара доста извори на информации со цел да може да донесе соодветна одлука.
2. *Одговор на повик во call центар* – многу call центри се потпираат на добри пребарувачки системи и бази на знаење со соодветна документација за успешно услужување на клиентите. Брзо пребарување и наоѓање на најрелевантните одговори со помош на напредно пребарување ќе придонесе услугата да ја врши помалку стручен персонал за пониска плата. Од друга страна повиците за поддршка ќе траат пократко и ќе може да се опслужат повеќе повици со истиот број на оператори.
3. *Продажба на постоечки клиент* – за успешна продажба на постоечки клиент навистина е важно добро познавање на неговите главоболки, навики, најкорисните контакти во компанијата и сл. Поради ова, еден успешен ЦРМ

систем потребно е да содржи напредно и ефикасно пребарување низ сите информации во него како и брз и лесен пристап (брзо пребарување) и приказ и анализа на постоечки информации за клиенти внесени во системот.

2.3 Пребарувачка машина

Модерните пребарувачки машини се највообичаениот начин за пристап до информации, пред сè поради огромната распространетост на WWW. Концептот на информација во овој контекст може да биде многу погрешен бидејќи е тесно поврзан со корисничкото барање. Пребарувачката машина е систем кој на корисничко барање, враќа резултати каде се наоѓа документот поврзан со даденото барање, и тоа во форма на патека до документ.

Во овој контекст, пребарувачката машина се занимава со проблемот на пронаоѓање и презентирање на документи со неструктурирани податоци кои го задоволуваат барањето за информација од самата колекција на документи (González, 2008). Пред да се продолжи понатаму добро е да се разјаснат неколку термини поврзани со пребарувачката машина. Најпрво поимот **неструктурирани** кој означува податоци кои имаат семантички гледано произволна структура, која може недвосмислено да биде пренесена на компјутер, за разлика од на пример релационите бази на податоци кои податоците ги чуваат структурирано во табели со еднозначни полиња. Поимот **документ** пред сè реферира на грануларноста на информацијата презентирана на корисникот и означува апстракт, статија, веб страница, секција од книга, е-маил, итн.. Односно, поимот документ не е ограничен само на текстуална информација, бидејќи корисниците може да се заинтересирани за пребарување или пристап до мултимедијални податоци, како видео, аудио или слики. Поимот **колекција** означува складиште на документи од каде се пребарува и враќа информацијата. **Корисничко барање за информација** означува пребарување кое мора да биде преведено со цел една пребарувачка машина да може да врати информации релевантни на барањето.



Слика 2.5. Поедноставена архитектура на пребарувачка машина (Inkpen, 2013)

Последно од ова разгледување на основната терминологија на пребарувачките машини е **презентацијата** на информацијата до корисникот. Презентацијата мора да ја прикаже информацијата на таков начин што би му олеснила на корисникот да пронајде документи за кои е заинтересиран. Добра пребарувачка машина е потребно да овозможи прелистување и филтрирање како опции за олеснување на работењето со вратените резултати.

Архитектурата на една пребарувачка машина е потребно да ги исполнува следните услови (Grossman, 2004):

- *Скалабилност.* Архитектурата мора да биде скалабилна за да може да опфати раст на колекцијата на документи.
- *Ефикасност на индексот.* Индексите мора да се градат во разумна временска рамка.
- *Ефикасност на пребарување.* Корисниците треба да бидат во можност да специфицираат колку време сакаат да чекаат за дадено пребарување. Секогаш постои баланс помеѓу колку долго е потребно да се обработи едно пребарување, и колку релевантни документи може да бидат пронајдени. Доколку времето за пребарување е неограничено, пребарувачката машина ќе може темелно да ги пребара сите информации и како резултат да ги најде совршените документи. Опцијата овој параметар да се дефинира согласно потребите на корисниците, ќе дозволи корисникот самиот да укаже на времето на кое е подготвен да чека за релевантните резултати.
- *Ефективност на пребарување.* Корисниците мора да се во можност да ги пронајдат документите кои се релевантни при даденото пребарување. Кај

пребарувачките машини е неопходно да постои високо ниво на ефективност.

Пребарувачката машина се состои од неколку процеси или фази кои обединуваат неколку различни компоненти. Најзначајни се: индексирање и scoring (Manning et al., 2009).

2.3.1 Индексирање

Процесот на индексирање е составен од неколку подпроцеси. Во првиот подпроцес се дефинира и гради речникот на зборови по направена обработка на колекцијата документи. Следно, од вака дефинираниот речник, се конструира индексот кој поради својата големина поминува низ дополнителен процес на компресија.

Двете главни компоненти на инвертираниот индекс се *речникот* и *постинг листите (posting lists).* При изградбата на инвертиран индекс најважни чекори се следните (Manning et al., 2009):

1. Собирање на документите кои е потребно да се индексираат;
2. Токенизација на текст, претворајќи го секој документ во листа на токени;
3. Лингвистичко претпроцесирање за креирање на листа на нормализирани токени кои преминуваат во зборови;
4. Индексирање на документите каде секој збор се појавува преку креирање на инвертиран индекс кој се состои од речник и постинг листи.

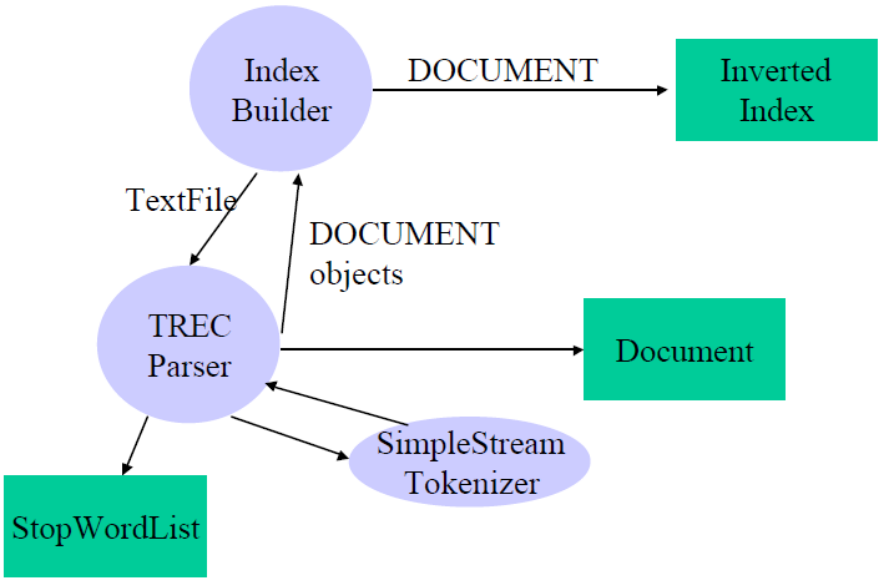
2.3.1.1 Веб crawler

Веб crawler-и популарно наречени пајаци или роботи, имаат задача да собираат веб страници за да ја изградат колекцијата на текстови за целиот систем на пребарувачката машина (Inkpen, 2013). Текстот се извлекува од HTML кодот на веб страниците, а може да се сочуваат и дополнително информации за форматот. Crawler-от започнува со една или повеќе http адреси (URL-а) и доколку се работи за веб страница,се обидува рекурзивно да ги следи линковите со цел да пронајде и дополнителни страници. Доколку не се работи за веб страница, Веб crawler-от може да ги добие линковите до документите кои е потребно да бидат дел од колекцијата документи и истите само да ги проследи за понатамошна обработка низ системот. Доколку се работи за веб страници, Веб crawler-от може да продолжи со depth-first пребарување, каде го следи првиот линк од страницата и сите линкови од новите страници до кои доаѓа преку тој линк, потоа се враќа назад и ги следи и преостанатите линкови од страницата. Друга опција би билobreadth-first пребарувањето каде ги следи сите линкови од една страница, потоа линковите од страниците кон кои покажуваат претходните страници.

Bread-first има предност бидејќи истражува униформно надвор од root страницата но бара поголема меморија за чување на сите линкови кои чекаат да бидат поминати од претходното ниво. Depth-first пребарувањето има потреба од помалку меморија но може да се загуби во следењето на еден единечен линк. Начинот на кој нови линкови се додаваат на редот зависи од пребарувачката стратегија. Хеуристичкото подредување на редот резултира во фокусиран crawler кој го насочува пребарувањето кон страници од интерес. Пајакот мора да внимава да не ги поминува повторно истите страници и линкови т.е. мора да води евиденција на страниците кои веќе ги има посетено. Одредени веб страници користат специјални robot exclusion протоколи кои се користат за да спречат истите да не бидат индексирани од Веб crawler-от.

2.3.1.2 Конструкција на индекс

Ниту една пребарувачка машина нема време за секвенцијални скенирања на колекцијата на документи. Наместо тоа се гради инвертиран индекс, структура која претставува мапирање на секој различен збор со листа на документи кои го содржат тој збор (Grossman, 2004). Вообичаено, градењето на индексот е интензивен пресметковен процес. Многу познати алгоритми постојат со цел да се намали времето потребно за изградба на инвертираниот индекс. Во основа овие алгоритми користат фиксна големина на меморијата и градат колку е возможно повеќе од инвертираниот индекс во меморија, а потоа продолжуваат со запишување на диск. На крај се креира едно множество од привремени датотеки кое понатаму се спојува во еден единствен инвертиран индекс.



Слика 2.6. Архитектура за конструирање на инвертиран индекс (Grossman, 2004)

Архитектурата која се однесува на конструирање на инвертираниот индекс се состои од следните елементи:

- **Документ**: Чува информации за еден документ. Парсерот креира документ објекти и клучните компоненти на секој од овие објекти се уникатните идентификатори на документот и листа од различни зборови кои се содржат во документот. За секој збор се чува и бројот на појавувања во документот.
- **Инвертиран индекс**: Овој објект ја содржи целата структура изградена за ефикасно пребарување на документи. Во основа, инвертираниот индекс се состои од речник на зборови, со слог за секој различен збор во колекцијата на документи. За секој збор, постои поинтер до поврзана листа од јазли кои ги содржат идентификаторот на документот и фреквенцијата на зборот во документот. Речникот се нарекува *индекс* додека поврзаната листа *posting list.*
- **Index Builder:** ова е објект кој го води процесот на конструкција на индексот.
- **PostingListNode:** претставува објект кој содржи еден запис од posting листата.
- **StopWordList:** објект кој чува листа на зборови кои е потребно да бидат игнорирани при изградбата на индексот, т.н. omitted words.

Дополнителни подобрувања на перформансите за време на процесот на индексирање се прават во процесот на парсирање. Често употребувани суфикси се отстрануваат со цел да се намали бројот на зборови кои во суштина означуваат ист збор, како и сврзници и многу општи зборови како “ако“, “ама“ и слично. На крај постојат и бројни алгоритми за компресија на инвертираниот индекс до 1/10 од неговата големина со цел не само да се намали потребниот простор за складирање, туку да се намали и бројот на I/O операции потребни за изградба и пребарување на индексот.

Во процесот на конструирање на индексот потребно е да се разгледаат пред сè податочните структури кои можат да подржат ваква индекс структура како и хардверските ограничувања (Manning et al., 2009).

Генерално постојат неколку начини за градење на индексот:

- Статична колекција
  - BSBI (Blocked Sort Based Indexing)
  - SPIMI (Single–pass–in–memory-indexing)
  - Distributed indexing (дистрибуирано индексирање)
- Динамичка колекција
  - Dynamic indexing (динамичко индексирање)

Статична колекција подразбира колекција на документи која ретко се менува. За овој тип на колекции на документи развиени се следните методи на индексирање: Blocked Sort-Based, Single-pass-in-memory и Дистрибуирано индексирање.

За мали колекции на документи доволно ефикасен метод е и конструирањето на непозиционирaн индекс каде што сите претходно наведени чекори за изградба на речникот и индексот може да се чуваат во меморија. Сепак кога се работи за поголеми колекции на документи, трите горенаведени методи се посоодветни бидејќи индексот неможе да се смести во главната меморија.

Динамичка колекција на документи соодветно претставува колекција која се менува често. Методи за поддршка на динамички променливи текстуални колекции може да бидат за внес и за бришење на документи.

Повеќето методи ја применуваат основната шема каде се одржува главен индекс на диск додека сите нови/променети/избришани документи се сместени во мал помошен индекс во меморија. Пребарувањето се прави низ двата индекса и се спојуваат резултатите. Периодично се спојува помошниот со главниот индекс откако ќе се искористат достапните ресурси во главната меморија или во даден временски интервал.

Хардверски основи

Бидејќи во изградбата на пребарувачка машина многу одлуки зависат од компјутерскиот хардвер кадешто ќе биде поставен системот, во ова поглавје на кратко се наведени некои значајни карактеристики кои е потребно да бидат земени во предвид при изборот на хардверска инфраструктура за изградбата на индексот (Manning et al., 2009).

1. Пристапот до податоци во меморија е многу побрз отколку пристапот на диск. Потребни се неколку clock cycles (можеби дури и 5 x 10^{-9} секунди) за пристап до бајт во меморија, но многу подолго за пренос на истите од диск (2 x 10^{-8} секунди). Во секој случај најдобро е колку што е можно повеќе податоци да се чуваат во меморија, особено податоците на кои е потребно да им се пристапува често. Оваа техника се нарекува кеширање.
2. При читање и запишување од диск, потребно е време за главата од дискот да се помести до специфичен дел од дискот каде податоците се лоцирани. Ова време се нарекува seek time и во просек се движи околу 5 ms за просечни дискови, но во овој временски период никакви податоци не се пренесени туку само

лоцирани. Со цел да се максимизира брзината на пренос на податоци, делови од податоците кои се читаат заедно (во мемориски блокови) потребно е да бидат чувани во близина т.е. еден до друг за да се намали што повеќе времето за пристап (движењето на главата од една до друга локација).

3. Оперативните систем генерално запишуваат и читаат податоци во цели блокови. Па така читање на еден бајт од диск зафаќа исто време како и читање на целиот блок. Делот во меморија каде се читаат или запишуваат цели блокови се нарекува бафер.
4. Пренесувањето на податоци од диск во меморија се прави преку магистралата, која го поврзува дискот со главната меморија. Ова значи дека процесорот е слободен за процесирање на податоци за време на I/O операциите. Овој факт може да се искористи за вршење на податочните трансфери во периоди кога процесорот е зафатен со други операции.
5. Серверите кои се користат за пребарувачките машини типично имаат неколку GB главна меморија но неретко и десетици и стотици GB. Расположивиот простор на дискот е значително поголем.

2.3.1.3 Речник на зборови

Речникот претставува централна податочна структура која се користи за управување со множеството на зборови кои се наоѓаат во колекцијата на текстови (Manning et al., 2009). Тој обезбедува мапирање од множеството на индексирани зборови до локацијата на нивните постинг листи. Прва операција при пребарување е лоцирањето на зборовите од барањето во постинг листите на индексот. Во време на индексирање, способноста на речникот за прегледување ѝ дозволува на пребарувачката машина брзо да ја добие мемориската адреса во инвертираната листа за секој влезен збор и да прикачи нов постинг на крајот од таа листа. Имплементациите на речник кои постојат кај пребарувачки машини најчесто ги поддржуваат следните операции (Buttcher et. al, 2010):

1. Внес на нова вредност за одреден збор.
2. Пронајди и врати ја вредноста за одреден збор.
3. Пронајди и врати ги вредностите за сите зборови кои започнуваат со одреден префикс.

При изградбата на индекс од колекцијата на текстови, пребарувачката машина

изведува операции од тип 1 и 2 кога е потребно да пребара влезни зборови во речникот и да додаде постинзи за овие зборови во индексот. Дури откако индексот е изграден, пребарувачката машина може да процесира пребарувања изведувајќи операции од тип 2 и 3 со цел да ги лоцира постинг листите за сите барани зборови. Иако операциите од тип 3 не се навистина неопходни, тие се корисна карактеристика бидејќи ѝ овозможуваат на пребарувачката машина да поддржи пребарувања со префикс од формата "inform*", споредување на сите документи кои содржат збор кој почнува со "inform".

За типична колекција на текстови од природен јазик, речникот е релативно мал во споредба со целосната големина на индексот. Примери на неколку вакви колекции на текстови се дадени во Табела 2.1.

Табела 2.1. Основни информации за колекции на текстови (Buttcher et. al, 2010)

	Shakespeare	**TREC45**	**GOV2**
Бр. на токени	1.3×10^6	3.0×10^8	4.4×10^{10}
Бр. на зборови	2.3×10^4	1.2×10^6	4.9×10^7
Речник (без компресија)	0.4 MB	24 MB	1046 MB
Docid речник	n/a	578 MB/200 MB	37751MB/12412 MB
Индекс на фреквенција	n/a	1110 MB/333 MB	73593 MB/21406 MB
Позиционен индекс	n/a	2255 MB/739 MB	245538 MB/78819 MB
Schema-ind. Индекс	5.7 MB/2.7 MB	1190 MB/532 MB	173854 MB/63670 MB

Доколку речникот е мал, тогаш истиот може целосно да биде вчитан во главната меморија на системот. Два најчесто користени начини за реализација на речници во меморија се:

- *Sort-based (Search trees) речник*, каде сите зборови кои се појавуваат во колекцијата се подредени во сортирана низа или пребарувачко дрво по азбучен ред. Lookup операциите се изведуваат преку премин на дрвото (tree traversal во случај на пребарувачко дрво) или бинарно пребарување (во случај на сортирана листа).

- *Hash–based речник* каде секој индексен збор има соодветна вредност во Хаш табела. Колизиите во Хаш табелата (т.е. два зборови да им се додели истата Хаш вредност) се разрешуваат преку т.н. Chaining - каде зборовите со истата Хаш вредност се поставуваат во поврзана листа.

Избор на единица за индексирање

Значаен чекор при изградбата на речникот на зборови од колекцијата на текстови е изборот на единицата за индексирање (Manning et al., 2009). До сега се претпоставуваше дека документите се фиксни единици во процесот на индексирање. На пример, секоја датотека во папка се зема дека е документ. Но во одредени случаи потребно е да се пристапи на друг начин. На пример традиционален Unix е-маил фајл чува секвенца на е-маил пораки во еден фајл, додека можеби е потребно секоја порака да се разгледува како посебен документ. Исто така, многу е-маил пораки содржат прикачени документи и можеби е потребно и сите прикачени документи и е-маил пораката да се гледаат како засебни документи. Доколку некоја порака содржи архивиран фајл, можеби ќе биде потребно истиот да се декомпримира и да се гледа секој фајл кој го содржи архивата како засебен документ. Во некој случај потребно е пак повеќе фајлови да се гледаат како еден документ. Генерално за многу долги документи се поставува проблемот со грануларноста на индексот. Така за колекција на книги би било лоша идеја да се индексира една цела книга како документ. Наместо тоа е подобро да се индексира секое поглавје или параграф како мини - документ. Но зошто да се запре овде, што ако секоја реченица претставува еден документ. Станува јасно дека овде се работи за баланс помеѓу precision и recall. Доколку единиците се многу мали, има големи шанси да се промашат значајни параграфи бидејќи зборовите се дистрибуирани низ повеќе документи, додека доколку единиците се преголеми тогаш ќе се добие поголем број на резултати низ кои корисникот ќе биде потребно да пребарува.

Токенизација

Следен чекор во процесот на изградба на речникот е Токенизација (Manning et al., 2009). За дадена секвенца на карактери и дефинирана документ единица, токенизација претставува задача на поделба на секвенцата во помали делови наречени токени, додека во меѓувреме се врши исфрлање на нерелевантни карактери и ознаки, како на пример специјалните знаци. Овие токени понатаму се нарекувани термови или зборови

но некогаш е важно да се направи соодветна разлика помеѓу типот и токенот. Токен претставува инстанца на секвенца од карактери од некој документ кои се групирани заедно во корисна семантичка единица за процесирање. Тип претставува класа на сите токени кои ја содржат истата секвенца на карактери. Збор претставува тип кој е вклучен во речникот на еден IR систем. Така на пример множеството на индексни зборови може да е комплетно различно од токените, каде индекс зборовите се добиваат преку стандардни техники за нормализација.

Проблемите со токенизација се специфични на јазикот. Поради ова е потребно и да се дефинира т.е. познава јазикот на документот. Со ова ќе може да се земат соодветни специфики за јазикот како апостроф, комплексни зборови и сл.

Стоп зборови

Покрај токенизацијата друг значаен процес е елиминирањето на најчестите зборови – стоп зборови (Manning et al., 2009). Понекогаш некои често користени зборови или сврзници кои се од мала вредност во пребарувањето и селекција на документи кои му се потребни на корисникот едноставно се исклучени од речникот во целост. Генералната стратегија за одредување на стоп листа е да се сортираат зборовите според нивната фреквенција во колекцијата и да се земат најчесто појавуваните зборови, и тоа може и рачно поради семантичката содржина која ја носат. Креирањето на стоп листа значително го намалува бројот на постинзи кои системот мора да ги чува. Во голем број на случаи неиндексирањето на стоп зборови не носи значителна штета, затоа што пребарувања на корисникот со или без сврзниците “и“ и “ама“ нема да биде многу различно. Сепак во некои случаи како пребарување фраза ова има големо влијание. Така, на пример, во фразата “Претседател на Соединетите Американски Држави“ сврзникот “на“ игра значајна улога додека во фразата “Претседател“ и “Соединетите Американски Држави “ сврзникот “и” нема таква улога.

Нормализација на зборови

Следен можен чекор во процесот на изградба на речникот на зборови е процесот на нормализација (еквивалентна класификација на зборови) (Manning et al., 2009). Најлесен случај во процесот на индексирање на еден документ и воедно пребарување на дадена секвенца е доколку поделениот документ, во токени, се совпаѓа со токените на пребарувачката сентенца. На пример, кога корисник ќе пребара со збор “USA” соодветно да направи match и со документи кои го содржат зборот “U.S.A.“. Процесот

на нормализација на токени претставува канонизација на токените со цел да се постигне совпаѓање дури и кога постојат мали и вештачки разлики во карактер секвенците на токените. Најчесто користениот начин за нормализација е имплицитно креирање на класи на еквивалентност, кои се најчесто именувани според еден член на множеството. Алтернатива на креирањето на класи на еквивалентност е одржување на врска помеѓу ненормализирани токени. Овој метод може да биде проширен со рачна конструкција на листа на синоними како “кола“ и “автомобил“. Овие врски помеѓу зборови може да бидат постигнати на два начина. Вообичаениот начин е да се индексираат ненормализираните токени и да се одржува проширена листа од повеќе вредности за даден збор од речникот, кои треба да се земат во превид при пребарувањето. Алтернативниот начин е да се изведе експанзија за време на конструирањето на индексот. Така кога документот содржи збор “автомобил“, истиот да се индексира и под “кола“.

Но мора да се наведе дека употребата на било кој од овие методи е далеку помалку ефикасна од употребата на класи на еквивалентност бидејќи и во двата случаи постојат повеќе постинзи кои е потребно да се чуваат. Од друга страна овие приоди се многу пофлексибилни во споредба со класите на еквиваленција бидејќи листите на експанзија може да се преклопат но повторно да не бидат идентични. Таков пример е зборот “windows“ кој во случај кога е напишан со голема почетна буква може да означува дека се пребарува Windows оперативниот систем но тоа не е точно во оваа ситуација.

Stemming & lemmatization

Друг чекор во процесот на креирање на речникот од зборови е процесот на stemming и lemmatization (Manning et al., 2009). Поради граматички причини, документите може да користат различни форми на зборот така на пример организира, организирам и организираат. Дополнително постојат и фамилии на зборови кои имаат иста деривација (основа) како зборовите демократија, демократски, демократизација. Во бројни ситуации се чини дека би било соодветно при пребарување на еден од овие зборови да се вратат сите документи кои содржат и друг збор од множеството на слични зборови.

Stemming претставува груб хеуристички процес кој ги отсекува краевите на зборовите во надеж дека ќе ја постигне својата цел во поголем дел од случаите и најчесто вклучува отстранување на деривациони афикси, како во случај на зборови “adapt“ и “adaptability“. Додека процесот на lemmatization најчесто се однесува на техники и методи за употреба на речник и морфолошка анализа на зборови, нормално со цел да се

отстранат само непотребни завршетоци и да се врати на коренското потекло на зборот т.н. *лема.* Двете техники доколку на пример се применат над зборот “saw“ истите ќе вратат различни резултати. Така процесот на stemming ќе врати можеби само s, додека lemmatization ќе се обиде да врати “see“ или “saw“ зависно од тоа дали токенот е во употреба на глагол или именка. Сепак анализа покажува дека за Европските јазици поголем придонес и подобрување носат техниките на stemming за разлика од lemmatization.

По завршување на овие чекори системот треба да ги извлече зборовите кои е потребно да бидат индексирани и да ги внесе во индексот.

2.3.2 Пребарувачки процесор

Пребарувачкиот процесор како компонента на пребарувачката машина вклучува едноставен GUI кој им овозможува на корисниците да внесуваат пребарување како листа од зборови без никакви Булови оператори (Grossman, 2004). Пребарувачкиот процесор ги зема зборовите за пребарување и враќа рангирана листа на документи кои се сметаат релевантни за барањето. Постојат бројни функции или мерки за сличност кои за дадена комбинација од пребарување - документ ја пресметуваат релевантноста на барањето за даден документ.

Идентификување на добри документи се нарекува *рангирање по релевантност (relevance ranking)* и постојат бројни методи за пресметување на релевантност. Архитектурата на пребарувачката машина е потребно да овозможи едноставно менување на мерката за сличност без фундаментални промени на инвертираниот индекс. Овие мерки е потребно да бидат земени како конфигурациски параметри на пребарувачката машина, дури и доколку е потребно за понапредни корисници да се овозможи избор во време на пребарување. Вистинската причина поради која релевантни документи не се пронајдени преку едноставна споредба на низи е разликата во јазикот (семантиката) на описот во релевантен документ и јазикот на изразување на корисникот при формирање на пребарување. Може како пример да се земе барањето “find documents about red cars” и документ кој опишува “rose-colored automobile”.

Стандардна евалуација на пребарувачки машини во последните десетина години покажува дека најдобрите пребарувачки алгоритми просечно пронаоѓаат 25-35% од релевантните документи. Во исто време постои широко уверување дека човек може да пронајде 70-80% од релевантните документи (човекот исто така прави многу грешки кога пребарува) (Grossman, 2004). Разликата во точноста постои бидејќи човек може

лесно да се префрли помеѓу јазикот на пребарувањето и јазикот на документот и ја прави оваа транслација многу поточно од алгоритам за рангирање кој едноставно само прави споредба на зборовите од барањето со зборовите во документот. За справување со овој проблем развиени се бројни експанзивни техники на пребарување кои на некој начин се обидуваат да додадат зборови на барањето со надеж дека истите ќе помогнат да се пронајдат релевантните и репрезентативните документи за даденото пребарување. Некои попознати техники од овој тип се thesauri, семантички мрежи и повратен одговор за релевантност (relevance feedback).

2.3.2.1 Scoring

Во процесот на пребарување на информации во еден IR систем крајниот резултат враќа листа на документи кои соодветствуваат со поставеното барање. IR системот не враќа експлицитни информации и одговори на прашања од корисникот, туку укажува на постоењето и локацијата на документи кои можеби ја содржат потребната информација (Hiemstra, 2009). Оваа листа се нарекува листа на релевантни документи. Совршен систем за пребарување секогаш е потребно да враќа релевантни резултати на пребарување по поставеното барање. Формирањето на рангирана листа на релевантни документи се случува во последниот чекор на споредба на информациите кои ги содржат документите во колекцијата со пребарувањето на корисникот. Идејата е најрелевантните документи да бидат помеѓу првите во вака рангираната листа со цел корисникот во најбрзо време да дојде до потребната информација. За таа цел постојат едноставни но ефективни алгоритми за рангирање кои ја користат дистрибуцијата на фреквенцијата на зборовите од документот, како и други статистички мерки над информациите кои ги содржи еден документ.

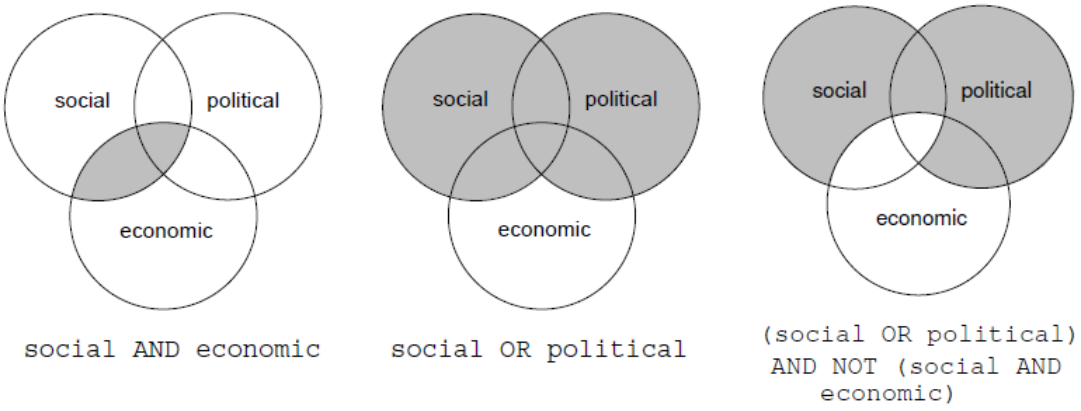
Во продолжение се наведени неколку значајни модели за рангирање на релевантни документи кај еден IR систем.

2.3.2.2 Boolean модел

Постојат повеќе различни модели за рангирање на документите по нивната релевантност за дадено пребарување од корисникот. Boolean моделот припаѓа на групата exact-match каде документите не се рангираат според релевантност туку според припадноста на даден збор кон даден документ (Manning et al., 2009). Овој модел е споменат во магистерскиот труд бидејќи е првиот модел кој се користел во IR системите но секако денес и критикуван поради својата ограниченост и застареност.

Овој модел може да биде објаснет со размислување за секој збор од упитот како

дефиниција на едно множество на документи. На пример зборот *економист* едноставно го дефинира множеството на сите документи кои се индексирани со тој збор. Со едноставна употреба на математичката логика на George Boole, пребарување зборови и трите основни оператори, логички продукт оператор AND, логички сумарен оператор OR и логички оператор за разлика NOT, се формираат множества на документи кои ги исполнуваат условите и се вратени од пребарувачката машина. Предности на Boolean моделот се чувството на контрола кое им го дава на понапредните корисници кои го користат. Со овој модел веднаш е јасно зошто даден документ не е или е вратен за дадено пребарување. Доколку множеството на документи е премало или преголемо, директно е јасно кои оператори можат соодветно да произведат поголемо или помало множество на документи.



Слика 2.7. Пример на логичките оператори на Boolean модел

За не толку напредни корисници моделот има значајни недостатоци. Главниот недостаток е што овој модел не обезбедува рангирање на вратените документи како резултати. Моделот или враќа или не враќа документ, што понекогаш значи системот да носи тешки решенија. Така на пример во случај на пребарувањето social AND worker AND union секако нема да врати индексирани документи со party, birthday или cake, но исто така нема да врати документи кои ги содржат social и worker но им недостига union. Повеќе од јасно е дека вториот документ е порелевантен за пребарувањето, но моделот нема начин како да направи разлика помеѓу двата.

2.3.2.3 Vector space модел

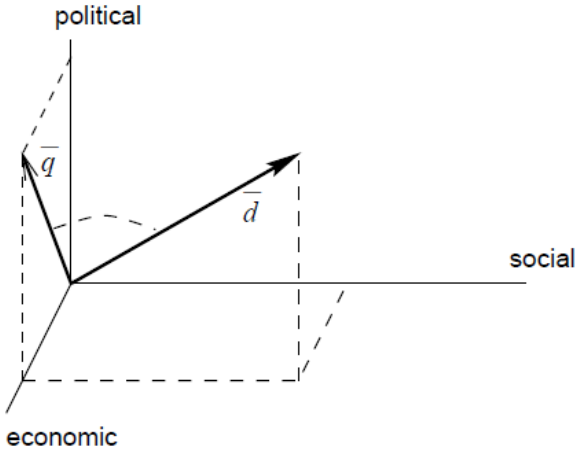
Според Hiemstra (2009), Peter Luhn е првиот кој воведува статистички начин за пребарување на информациите. Со цел да се пребара колекцијата на документи корисникот најпрво мора да подготви документ кој е сличен на документите кои се потребни. Нивото на сличност помеѓу подготвениот документ и колекцијата на

документи се користи за рангирање на добиените резултати.

Salton и неговите колеги предложиле модел базиран на Luhn критериумот на сличност кој има посилна теоретска основа (Salton & McGill, 1983). Тие ги земаат во предвид репрезентацијата на индексот и упитот како вектори вградени во повеќе димензионален Евклидов простор, каде на секој збор му е доделена посебна димензија. За мерка за сличност вообичаено се користи косинус од аголот кој ги разделува двата вектори $\vec{d}$ и $\vec{q}$. Косинус од аголот е 0 кога векторите се ортогонални во повеќе димензионалниот простор додека е 1 кога аголот е 0 степени. Косинусната формула е прикажана во продолжение:

$$score(\vec{d}, \vec{q}) = \frac{\sum_{k=1}^{m} d_k . q_k}{\sqrt{\sum_{k=1}^{m}(d_k)^2} . \sqrt{\sum_{k=1}^{m}(q_k)^2}}$$

Метафората од аглите помеѓу векторите во повеќе димензионален простор го прави поедноставно објаснувањето на моделот на нестручни лица. Во продолжение е наведен графички пример на претставување на истиот пример од зборови од претходното поглавје за Boolean моделот. До три димензии е доста лесно да се визуализира документот и пребарувачките вектори. Мерењето на косинус од аголот помеѓу векторите е еквивалентно на нормализација на векторите на единица должина и пресметка на нивниот векторски производ.



Слика 2.8. Графички приказ на Vector space модел

Доколку индексите и пребарувањата се правилно нормализирани тогаш формулата на Luhn преминува во следната форма:

$$score(\vec{d}, \vec{q}) = \sum_{k=1}^{m} n(d_k).n(q_k) \text{ каде } n(v_k) = \frac{v_k}{\sqrt{\sum_{k=1}^{m}(v_k)^2}}$$

Постојат неколку добро поставени vector space модели меѓу кои во овој труд на кратко ќе се опише моделот на Joseph Rocchio (Manning et al., 2009). Rocchio го претставил следниот алгоритам за повратна спрега на релевантност (relevance feedback), каде $\vec{q}_{old}$ е оригиналното пребарување, додека $\vec{q}_{new}$ е ревидираното пребарување, $\vec{d}_{rel}^{(i)}(1 \leq i \leq r)$ претставува еден од r документите кои корисникот ги избрал како релевантни и $\vec{d}_{nonrel}^{(i)}(1 \leq i \leq n)$ е еден од n документи кои корисникот ги избрал како нерелевантни.

$$\vec{q}_{new} = \vec{q}_{old} + \frac{1}{r}\sum_{i=1}^{r} \vec{d}_{rel}^{(i)} - \frac{1}{n}\sum_{i=1}^{n} \vec{d}_{nonrel}^{(i)}$$

Нормализираните вектори од документи и пребарувања може да бидат како точки во хиперсфера на единица должина од центарот. Во последната равенка, првата сума го пресметува центроидот од точките од познатите релевантни документи на хиперсферата. Во центроидот, аголот со познатите релевантни документи е минимизиран. Втората сума го пресметува центроидот од точките од познатите нерелевантни документи. Поместувајќи го упитот накај центроидот од познатите релевантни документи и настрана од центроидот од познатите нерелевантни документи гарантирано ги подобруваат перформансите на пребарувањето.

Главниот недостаток на Vector space моделот е што на никаков начин не дефинира кои вредности треба да ги имаат векторските компоненти. Проблемот со доделување на соодветни вредности на векторските компоненти е познат како term weighting. Salton и Yang покажале дела term weighting не е воопшто тривијален проблем и предложуваат т.н. tf.idf тежини, кои претставуваат комбинација од фреквенцијата на зборот tf и idf, инверзната фреквенција на документот која претставува вредност поврзана инверзно за фреквенцијата на документот df, која пак го претставува бројот на документи кои го содржат зборот.

Дополнителен проблем со vector space моделот е неговата имплементација. На пресметката на косинусот ѝ требаат вредности за сите векторски компоненти, но сите овие не се достапни во инвертираниот фајл. Во пракса се користат нормализирани вредности и алгоритам за векторски производ. Во ваков случај или нормализираните тежини се чуваат во инвертираната датотека или нормализираните вредности се чуваат посебно. И двете се проблематични во случај на почесто ажурирање на индексот. Така на пример додавање на еден нов документ ги менува фреквенциите на документите на зборовите кои ги има и во тој документ, што дополнително ги менува должините на

векторите на секој документ кој содржи повеќе од еден од овие зборови.

За разлика од методот на Rocchio, методот на k-најблизок сосед односно kNN класификациски метод ја одредува локално границата на одлука. Така за 1NN секој документ се доделува на класата од својот најблизок сосед. За kNN секој документ е доделен на мнозинската класа од неговите k најблиски соседи каде k е параметар. Рационалноста позади kNN класификацијата се базира на хипотезата за непосредна близина, каде се очекува тест документ d да ја има истата лабела како и тренинг документите лоцирани во локалниот регион околу d. Овој метод не е детално разгледан во магистерскиот труд.

2.3.2.4 Модел на веројатности

Во претходните модели, Boolean и vector space, споредбата на документи се прави на формални но сепак непрецизни пресметки на индексните зборови. За дадено пребарување, еден IR систем нема јасно разбирање за информацискатa потреба на упитот. Со дадено пребарување и репрезентација на документот, системот има несигурен погодок дали дадена содржина на документ е релевантна на информацискатa потреба. Теоријата на веројатност дава начин на размислување и одлука под овој тип на несигурност (Manning et al., 2009). Постојат повеќе модели за пребарување на информации кои имаат теорија на веројатност во својата основа. Во оваа магистерска теза преставен е OkapiBM25 моделот поради својата успешна тежинска шема и масовно користење.

Неколку приоди кои се обидуваат да го одредат процесот на дефинирање на тежински вредности на зборовите поформално се базирани на теоријата на веројатност. Поимот за веројатност на нешто, на пример веројатност за релевантност означен како P(R), е вообичаено формализиран преку концепт на експеримент, каде експеримент е процесот преку кои се врши набљудувањето. Множеството на сите можни излези од експериментот се нарекува sample space. Во овој случај sample space на P(R) може да биде {relevant, irrelevant}, и може да се дефинира случајна променлива R која ќе зема вредност {0, 1}, каде 0=irrelevant додека 1=relevant.

Доколку се дефинира експеримент каде се зема еден документ од колекцијата по случаен избор тогаш доколку го знаеме бројот на релевантни документи во колекцијата на пр. 100, и го знаеме вкупниот број на документи во колекцијата, на пр. 1 милион, односот помеѓу овие две ја дефинира веројатноста за релевантност:

$$P(R = 1) = {}^{100}/_{1000000} = 0.0001$$

Понатаму да се претпостави дека $P(D_k)$ ја претставува веројатноста дека документ содржи збор k со sample space {0,1}, каде 0=документ кој не го содржи зборот k и 1=документ кој го содржи зборот k, тогаш може да се искористи $P(R,D_k)$ за означување на заедничка веројатност на дистрибуција (joint probability distribution) со излези {(0,0), (0,1), (1,0), (1,1)} додека ознаката $P(R|D_k)$ за означување на условна веројатност на дистрибуција со излези {0,1}. Значи $P(R=1|D_k=1)$ е веројатноста за релевантност доколку се земат во предвид документите кои го содржат зборот k.

OkapiBM25: небинарен модел

За потребите на модерните текстуални колекции за пребарување, потребен е модел кој се фокусира на фреквенцијата на зборовите и должината на документот. Тежинската шема на BM25 уште и наречена Okapi weighting, е развиена како веројатностен модел чувствителен на овие вредности но без воведување на бројни дополнителни параметри на моделот (Manning et al., 2009). Наједноставното рангирање на документот d е претставено со следната формула:

$$RSV_d = \sum_{t \in q} log \frac{N}{df_t}$$

Оваа формула може да се прошири и да ја вклучи фреквенција на секој збор како и должината на документот:

$$RSV_d = \sum_{t \in q} log \left[\frac{N}{df_t}\right] \cdot \frac{(k_1 + 1) tf_{td}}{k_1 \left((1 - b) + b \times \left(\frac{L_d}{L_{ave}}\right)\right) + tf_{td}}$$

Овде, tf_{td} ја претставува фреквенцијата на зборот t во документот d и L_d и L_{ave} се должината на документот d и просечната должина на документите за целата колекција соодветно. Променливата k1 е позитивен подесувачки параметар кој врши калибрација на скалирањето на фреквенцијата на зборовите во документот. Вредност 0 за k1 соодветствува на бинарниот модел, додека голема вредност одговара на употреба на груба фреквенција на зборот. b исто така претставува подесувачки параметар $(0 \leq b \leq 1)$ кој го одредува скалирањето по должина на документ: b=1 означува целосно скалирање на тежината на зборот според должината на документот, додека b=0 соодветствува на немање нормализација на должината. Доколку и упитот е предолг тогаш може да се примени слична техника на поставување на тежини на пребарување на зборовите. Вредностите на овие параметри е потребно да бидат добиени со анализа и тестирање со помош на развојна тест колекција (рачно или преку

оптимизациски методи) и потоа вака добиените вредности да се тестираат над вистинската тест колекција. Експериментите покажале дека најсоодветно е вредноста на k1 да биде помеѓу 1.2 и 2 додека на b=0.75.

2.3.2.5 Јазичен модел

Јазичните модели почнале да се употребуваат во IR системи во доцните 90-ти. Тие потекнуваат од веројатностни модели за генерирање на јазици за системи за автоматско препознавање на говор во раните 80-ти. Автоматските системи за препознавање на говор комбинираат веројатности од два различни модели: акустичниот и јазичниот модел (Manning et al., 2009). Така акустичниот модел креира кандидати текстови во опаѓачки редослед на веројатност додека јазичниот модел одредува дали кандидатот е веројатна појава во јазикот кој се анализира.

Во еден IR систем јазични модели се креираат за секој документ. Со следење на овој приод, јазичниот модел на пример за книга за IR системи ќе додели висока веројатност за зборот "retrieval" укажувајќи дека таа книга е добар кандидат за превземање доколку упитот содржи збор "retrieval". Дадено дека документ D е релевантен, корисникот може да формулира пребарување со употреба на зборот T со веројатност P(T/D). Веројатноста се дефинира од текстот во документите. Така доколку документ содржи 100 зборови, и од тие на пример зборот "добар" се појавува два пати, тогаш веројатноста за "добар" дека документот е релевантен е 0.02. За пребарувања со повеќе зборови, се претпоставува дека зборовите се генерирани независно еден од друг т.е. дека условните веројатности на зборовите T1, T2, … за даден документ се множат:

$$P(T_1, T_2, \ldots | D) = \prod_i P(T_i | D)$$

Овој модел доделува веројатност нула за документи кои не ги содржат сите зборови. За некои апликации ова не е проблем како на пример веб пребарувачките машини каде пребарувањата се најчесто кратки и ретко се случува страница да ги содржи сите барани зборови, но за многу апликации знаат да се случат почесто празни резултати.

Поради ова се користи техника наречена smoothing или мазнење која доделува не нулти веројатности за невидени настани.

Постојат повеќе јазични модели кои може да се употребат во имплементации на IR систем како Униграм и Биграм моделот како и многу дополнителни покомплексни јазични модели како веројатностни context-free граматики кои се витални за задачи како препознавање на говор, корекција на спелување и машински превод каде е потребна веројатност во зависност од контекстот.

2.4 Трендови и имплементации

Денешните водечки светски софтверски компании произведувачи на ЕРП решенија се повеќе започнаа со воведување на напредни пребарувачки машини во своите системи со цел подобрување на можностите за лоцирање и пребарување информации низ нивните системи, а во корист на своите корисници. Дел од нив започнаа соработка со постоечки компании кои изработуваат вакви системи за интеграција на нивните апликации за поефикасно пребарување. Така Ziliak (2013) коосновач на xkzero, укажува дека според истражување спроведено од нив, корисниците на традиционалните ЕРП системи може да поминат и до 25% од своето време во пребарување на основни податоци низ своите системи. Нивниот систем GetX (Ziliak, 2013) овозможува пребарување во стилот на Google за познатите Sage ЕРП системи преку пребарување на индексирана база на податоци.

Дополнително, Google минатата година ја објави својата комерцијална алатка за пребарување која е наменета за компаниите за пребарување на информации кои се зачувани на нивните компјутери. Google Search Appliance (GSA) претставува надградувачка компонента кој обезбедува сервис за индексирање на документи кој им овозможува на компаниите да дојдат до податоци сочувани на различни локации, десктопи, интранети или архивски сервери. (Okuttah, 2013).

Оваа година беше претставена и нова фајл пребарувачка машина[1] која им овозможува на корисниците да пребаруваат било какви податочни датотеки. Овие пребарувачки машини се различни од стандардните со тоа што тие пребаруваат фајлови со употреба на свој crawler и тогаш додаваат линкови за симнување за документи, видеа, софтвер, музика и многу повеќе во нивната база на податоци. Резултатите од пребарувањето се релевантни на бараната содржина. Корисниците се во можност да ја пронајдат содржината која ја бараат без разлика дали се наоѓа на одреден сајт или е зачувана некаде низ Вебот. Дневните промени на сајтовите се ажурират во податоците на пребарувачката машина па корисникот добива ажурни информации.

Дополнително, се повеќе компании ги препознаваат предностите од ваква надградба на својот ЕРП систем. Таков пример е компанијата EnPro (Frey, 2013). Компанијата решава да имплементира скалабилно и аналитичко решение кое ќе обезбеди преглед во корисничките податоци во реално време. Со помош на in-memory технологиите овој

[1] http://www.filesclip.com/

приод овозможува податочно процесирање и аналитика во неколку секунди – користејќи машини од најнова генерација.

2.4.1 Промена на традиционалниот модел на перцепција на пребарување

Пребарувањата на корисниците се област која се повеќе е актуелна во SEO заедницата. Пред се забележана е промена на традиционалниот модел на перцепција на пребарувањето за крајниот корисник (Antony, 2013). Традиционалниот приод кон пребарување е нешто од типот "london tube stations". Ова претставува пребарување базирано на клучни зборови кое е тип кој досега ги задоволуваше пребарувачките потреби на корисниците. Сепак напредните пребарувачки машини повеќе не го разгледуваат единствено експлицитниот аспект на упитот, туку и имплицитниот, каде може да се вклучат во превид повеќе други информации како на пример iPhone корисник кој се наоѓа на улиците во Лондон. На овој начин резултатите од пребарувањето ќе ги вклучат и контекстуалните информации од корисникот. Изворот на имплицитниот дел од упитот е секако контекстот. Контекстот претставува информации кои го дополнуваат пребарувањето со информации за корисникот кои овозможуваат прилагодување и персонализирање на вратените резултати. Така може да се земат во предвид две интересни теми во SEO заедницата (Antony, 2013):

1. **Мобилно пребарување.** До сега ова го означуваше единствено уредот кој го користи корисникот. Но денес се повеќе луѓе пребаруваат од нивните смартфони додека се во канцеларија, дома или било каде и користат лаптопи и таблети во движење. Мобилното пребарување треба да го анализира корисникот и неговата состојба, дали е во движење или не, дали се наоѓа во својата канцеларија, на состанок, дома или во ресторан.
2. **Персонализирано пребарување.** Кога се споменува персонализирано пребарување најчесто се мисли на корисничките потреби и желби генерирани од социјални мрежи, пребарувачка историја, и сл. И додека генерирање на истото персонализирано пребарување во различни делови од денот и од различни локации, враќа различни резултати сепак на денешниот корисник сеуште не му е јасно зошто тие резултати се различни.

Покрај овие два примера, постојат и други области кои придонесуваат во персонализација на пребарувачките резултати, и сите тие аспекти може да се наречат контекст. Контекстуалната информација ја опфаќа мобилната и персоналната, како и цело множество на други сигнали. Имплицитниот контекст на упитот доаѓа целосно од

корисничкиот контекст. Така на пример веќе е најавен нов Android API кој вклучува превземање на информации од контекстуална природа, каде на пример од тие што развиваат Андроид апликации се бара да ја побараат информацијата од телефонот дали корисникот е во движење, вози велосипед и сл. Дополнително Google пред извесно време го купи Behavio, тимот позади Funf, "Social and Behavioural Sensing Framework". Оваа рамка во основа се обидува да го предвиди корисничкото однесување базирано на претходни и моментални состојби преку различни сензори од нивниот телефон.

2.4.2 Примена на IR системи

IR системите најпрво биле развиени за да помогнат во управувањето на огромните количества на научна литература и тоа уште од 1940-та година (Mihaescu, 2009). Многу универзитетски, компаниски и јавни библиотеки денес користат IR системи за да обезбедат пристап до бројни книги, весници и други документи. Комерцијалните IR системи нудат бази на податоци кои содржат милиони документи од најразлични области. Овие системи се особено корисни во многу различни области како автоматизација во компаниското работење и софтверското инженерство. Воедно секоја дисциплина која се потпира на документи за да ги извршува своите работни задачи може да има потенцијална корист од IR системите. Денес, IR системите имаат бројни апликации, се користат за пребарување на документи, односно содржини, метаподатоци на документи во рамки на традиционални релациони бази на податоци и интернет документи на многу посоодветен начин и овозможуваат намалување на работата и потрошеното време за пристап до одредена информација. Најдените документи како резултат на пребарувањето потребно е да бидат релевантни на бараната корисничка информација. Многу проблеми во IR може да бидат гледани како проблеми со прогноза т.е. предвидување на вредности за рангирање или рејтинг на веб страници, документи, музика и сл. и дополнително учење кој тип на информации ги пребарува дадениот корисник и какви се неговите интереси.

2.4.2.1 Дигитални библиотеки

Дигитална библиотека претставува колекција на документи чувани во дигитална форма и пристапна преку компјутер (Mihaescu, 2009). Дигиталната содржина може да биде чувана локално, или да се пристапува до истата преку компјутерските мрежи. Дигитална библиотека е една форма на IR систем. Многу академски библиотекари се активно вклучени во изградба на складишта за институционални книги, статии, тези и други информации кои може да бидат дигитализирани или се достапни во таква форма.

Многу од овие складишта се достапни за јавноста со неколку рестрикции поврзани со правото за отворен пристап до информации, за разлика од истражувачките публикации во комерцијални журнали каде постои ограничен пристап. Институционални, бесплатни и корпоративни складишта најчесто се реферираат како дигитални библиотеки.

Традиционалните библиотеки се првите институции кои започнале со користење на IR системите со цел да креираат каталози на записи за материјалите од библиотеката. Каталозите понатаму можат да бидат користени за онлајн или офлајн пребарувања од страна на корисници. Модерните библиотеки се трансформирани во дигитални како резултат на порастот во електронското издаваштво каде информацијата е достапна во дигитална форма. Преку употреба на Интернетот обезбеден е пристап до локални ресурси како и далечински пристап до бази на податоци во разни научни полиња и бизнис вклучително и текстуални журнали, весници и директориуми. Постојат бројни проекти за интернационални или национални дигитални библиотеки. Еден таков пример е Digital Libraries Initiative (DLI) поддржана од National Science Foundation (NSF), Department of Defense Advanced Research Projects Agency (DARPA) и National Aeronautics и Space Administration (NASA).

2.4.2.2 Пребарувачки машини (текст + слики)

Пребарувачка машина е една од најпрактичните апликации на IR техниките за големи колекции на текстуални документи (Mihaescu, 2009). Веб пребарувачките машини се најдобро познати примери, но постојат и многу други како: Десктоп пребарување, Компаниско пребарување (enterprise search), Федеративно пребарување, Мобилно пребарување, Социјално пребарување, итн. Веб пребарувачка машина е дизајнирана за пребарување на информации на WWW. Резултатите од пребарувањето се најчесто презентирани во форма на листа и се наречени hits. Информацијата може да се состои од веб страници, слики, информации и други типови на фајлови. Некои пребарувачки машини пребаруваат податоци достапни во бази на податоци и отворени директориуми. За разлика од Веб директориумите кои се одржувани од луѓе, пребарувачките машини оперираат со помош на алгоритми или се мешавина од алгоритамско и човечко едитирање.

Повратна информација за релевантноста е многу значајна тема кај IR особено во веб пребарувањето. Често присутен проблем е дека пребарувачките зборови во барањето се двосмислени и поради ова документи од друг нерелевантен контекст се вратени како

резултати или не е познато кои зборови го опишуваат точно бараниот проблем. Идејата за повратна информација за релевантноста им овозможува на корисниците да рангираат вратени документи како повеќе или помалку релевантни и на овој начин полесно, побрзо и поефективно да ги пронајдат потребните документи. Овие нови идеи се усвоени во image retrieval (пребарување на слики) бидејќи сликите тешко се пребаруваат со опис во зборови. Систем за пребарување на слики претставува компјутерски систем за прелистување, пребарување и враќање на слики од голема база на дигитални слики. Најтрадиционалните и основни методи за пребарување на слики употребуваат некаков метод на додавање на мета податоци како наслови, клучни зборови или описи на слики со цел пребарувањето да биде изведено над овие информации. Рачно означување на слики одзема многу време и носи дополнителни трошоци, па интересот за автоматското означување на слики покрена голем интерес во последно време кај научната заедница. Дополнително, експанзијата на социјалните апликации и семантичкиот веб се дополнителен импулс за развој на веб - базирани анотациски алатки за слики.

2.4.3 Постоечки пребарувачки машини и технологии

Според Sun (2009) популарните пребарувачки машини имплементирани како напредни IR системи со најактуелните техники и дизајн, постојано се користат од корисниците. Во основа овие пребарувачки машини со помош на графички кориснички интерфејс ги примаат пребарувањата од корисниците, и ги презентираат резултатите и рангирањето на истите од базата на документи, додека веб страниците се процесираат со робот и се индексираат и чуваат заедно со клучните зборови кои документот ги содржи. Генерално, успехот на пребарувачките машини се должи на нивната способност да ги рангираат резултатите. Способноста да се рангираат страници од вратените резултати од неколку милијарди страници е основен атрибут за успех.

2.4.3.1 Google

Основачите на Google, Сергеј Брин и Лери Пејџ на Универзитетот Стенфорд ја имаат креирано најпознатата пребарувачка машина за веб (Wills, 2006). Првичниот назив на пребарувачката машина бил BackRub, но подоцна бил променет во Google кој произлегува од зборот googol кој збор го означува број 10^{100}. Со финансиска поддршка на мала група на инвеститори, Брин и Пејџ ја основале Веб пребарувачката корпорација Google inc. во Септември 1998. Клучната технологија на Google претставува равенка за рангирање на страницата развиена од основачите. Иако повеќе

фактори влијаат на рангирањето на резултатите, сепак Google и самиот стои на ставот дека срцето на неговата пребарувачка машина е PageRank алгоритамот. Основниот принцип е дека системот треба во вратените резултати да вклучи веб страници кои високо рангираат во однос на поставеното барање за пребарување, доколку страницата е поврзана со многу други страници. Бизнис и академската заедница се свесни за популарноста на Google пребарувачката машина па максимизирањето на PageRank резултатот за даден веб сајт стана една од најважните компоненти од компаниските маркетинг стратегии.

Google континуирано го надополнува својот алгоритам за пребарување, а една од последните надградби е Hummingbird[2] алгоритамот за пребарување, кој според досегашните тврдења би требало да ги подобри сегашните пребарувачки резултати на Google. Употребата на Hummingbird со преостанатите компоненти покажува дека Google направи промена на стариот со нов мотор давајќи нова димензија на брзината и прецизноста во својата пребарувачка машина. Оваа надградба на Google е со цел да се подготви пребарувачката машина за имплицитните значења на секое пребарување, со повеќе акцент на секој збор од пребарувањето и неговото можно значење. Иако Google веќе имаше имплементирано нешто од овој тип во својата машина со Knowledge Graphs, сепак Hummingbird е наменет да ја искористи технологијата за процесирање на милијардите страници на Веб и заедно со Knowledge Graphs да придонесе за подобри резултати (Sun, 2009).

PageRank технологија

Успехот на Google, како што е наведено и погоре, се должи на начинот на кој Google ги рангира страниците, односно рангирачкиот алгоритам кој ја мери важноста на една веб страница. Како што и самите основачи изјавиле во 1998 за алгоритамот PageRank, се претпоставува дека страница A има страници T1, …, Tn кои покажуваат на неа како цитати, линкови и слично (Sun, 2009). Параметарот d е фактор на амортизација кој добива вредност помеѓу 0 и 1. Исто така постои функција C(A) која означува колку линкови излегуваат од страницата A. На тој начин PageRank за страницата A е следен:

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)})$$

Мора да се наведе дека рангирањата на сите страници формираат веројатностна

[2] http://searchengineland.com/google-hummingbird-172816

дистрибуција над сите страници, па сумата на сите рангови на страниците изнесува 1. PageRank-от или PR(A) може да се пресмета со употреба на едноставен итеративен алгоритам и соодветствува на основниот eigenvector од нормализираната линк матрица од вебот. Истиот целосно зависи од структурата на WWW.

2.4.3.2 Lucene

Lucene претставува open-source множество од библиотеки кое овозможува поддршка за развој и имплементација на мини апликации за пребарување низ документи[3]. Lucene воглавно се користи за пребарување низ текстуални содржини. Целиот пакет е збир од Јава библиотеки кои содржат функции за индексирање и пребарување на текстуални содржини. Иницијално беше достапна како open-source проект на Source Forge развиен од Doug Cutting во 1996, додека во 2001 стана дел од Apache Software фондацијата на Џакарта фамилијата како бесплатен Јава продукт. Поради своите моќни функционалности, Lucene наоѓа огромна примена при развој и имплементација на пребарувања во различни системи, како што тоа го направиле FedEx, Overture, Mayo Clinic, Epiphany и сл. Lucene содржи библиотеки кои може да се користат за вградување на едноставно индексирање и пребарување користејќи различни програмски јазици, како Delphi, Perl, C#, C++, Python, Ruby, PHP, итн. (Sun, 2009)

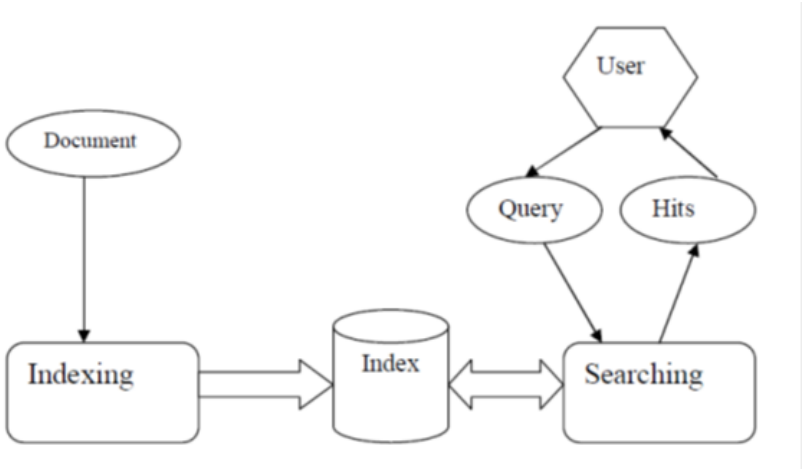
Основни Lucene поими (објекти) се:

- Document (Документ) – претставува еден запис во Lucene индексот. Овој објект претставува поврзана листа од Field објекти.
- Field (Поле) – За секој објект Document се чува листа од објекти Field кои се дефинираат во моментот на индексирањето. Овие полиња се всушност атрибутите на објектот Document кои Lucene треба да ги распознава, како на пр. назив, содржина, датум на креирање, автор, url, краток опис и сл.
- Term (Збор) – е основна Lucene единица, т.е. еден збор од содржината во Field објектите.
- Analyzer (Анализатор) – е компонента т.е. објект кој се користи за обработка на содржината во Field полињата, односно упитот при индексирање, или пребарување, соодветно. Оваа компонента може да се параметризира да користи различни стоп зборови кои не треба да се земаат во предвид при индексирање и пребарување. Истата може да се прилагоди за различни јазици.

[3] http://lucene.apache.org/

- Stemmer – е компонента т.е. објект дел од Lucene кој се користи за сведување на секој збор во неговата основна форма – корен на зборот. Ваквите компоненти се потпираат на комплексни алгоритми кои варираат за секој различен јазик и треба да ги опфаќаат сите граматички правила на еден јазик.
- TermDocs – е објект кој се користи за да се идентификува кои Term објекти ги содржи даден Document објект.
- TermFreqVector – е објект кој се користи за чување на фреквенцијата со која даден Term се појавува во конкретен Document објект.
- Directory – е објект со кој се креира директориумот во кој физички ќе биде лоциран индексот, односно се иницијализира директориум објектот низ кој ќе се се пребарува.
- IndexReader – е објект со кој се исчитува индексот сместен во даден Directory објект.
- IndexWriter – е објект кој се користи за запишување на Document записите во даден Directory објект.
- IndexSearcher – е објект кој со помош на IndexReader пребарува низ даден индекс.
- Query – е објект кој ги содржи клучните зборови внесени од страна на корисникот при пребарување.
- QueryParser – е објект кој има методи за парсирање на Query објектот внесен при пребарување користејќи различни Analyzer објекти.

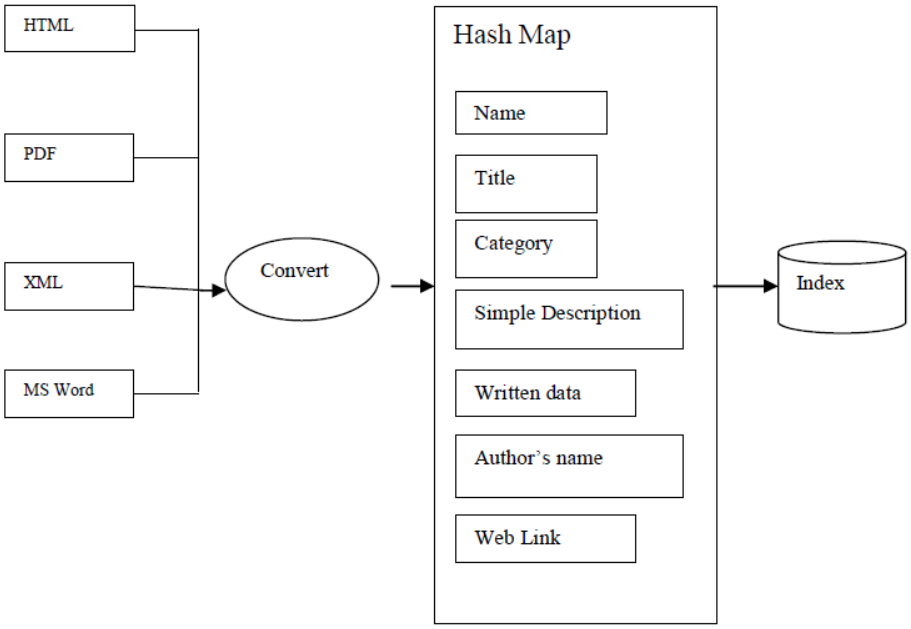
Ова се основните Lucene поими кои се користат во процесите на индексирање и пребарување. Целиот тек на активности, односно процес, почнувајќи од индексирање до приказ на резултати од пребарување е прикажан на Слика 2.2. Во продолжение се опишани различните активности кои се дел од овој процес.



Слика 2.2. Lucene текст базирана пребарувачка машина

Индексирање во Lucene

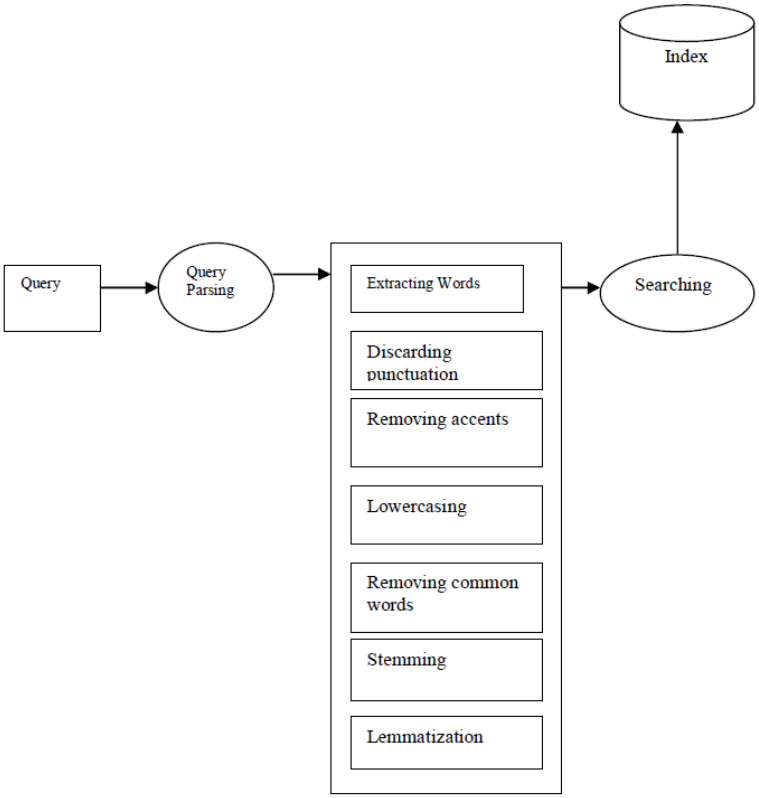
Со Lucene, процесот на индексирање се состои од конвертирање на сите различни датотеки во документ објекти со назив, наслов, категорија, опис, датум, автор и веб линкови. Овие атрибути се само пример за информации кои може да се чуваат за колекцијата на документи. При индексирање со Lucene може да се дефинираат различни атрибути според потребите на корисникот за кој се имплементира пребарувањето. За разлика од Хаш мапа, како индекс листа се користи векторска структура каде зборовите и документите може да се чуваат и да се прошират во едноставни секвенцијални структури. Кога се креира документ објект, индексерот го запишува документот во индекс фајлот. Се креира анализатор кој специфицира во кој директориум индексот физички ќе се наоѓа, дали ќе се креира нов индекс или се користи постоечки, оптимизација, кој модел за бодирање (scoring) ќе се користи и сл. Процесот на индексирање кај Lucene е прикажан на Слика 2.3.



Слика 2.3. Процес на индексирање на Lucene

Пребарување со Lucene

При пребарување, најпрво, упитот се парсира и конвертира во листа од обичен текст. Следно, анализаторот го процесира истото, при што се отстрануваат интерпункциските знаци, акценти, непотребни зборови и сл. Анализаторот со помош на Stemmer ги добива корените од зборовите во Упитот кои понатаму се користат за пребарување. Во исто време се креира објект во кој се вчитува директориумот во кој се чува индексот, и соодветно IndexReader и IndexSearcher објектите. IndexSearcher објектот со помош на search методот кој како аргументи ги прима Query објектот и празна колекција од документи, пребарува низ индексот со што се полни колекцијата на документи. Редоследот во кој се полнат документите зависи од бодирањето на истите кое пак зависи од т.н. scoring модел кој се избира и поставува при индексирање и пребарување на документи. Од колекцијата на документи се добива низа од документи т.н. hits. На Слика 2.4. е илустриран процесот на пребарување во Lucene.



Слика 2.4. Процес на пребарување на индекс во Lucene

Scoring на документи

Lucene scoring, односно бодирањето на документите е една од основните причини за да се избере Lucene како библиотека за имплементација на пребарувачка машина во било која апликација[4]. Lucene овозможува брзо сортирање на резултатите од дадено пребарување, истовремено криејќи ја комплексноста на целиот процес со користење на готови класи и методи. Во претходните сегменти опишавме неколку методи кои се користат при процесот на scoring и можеме да заклучиме дека истите се доста комплексни. Со помош на Lucene нема потреба од детално познавање на алгоритмите за сортирање на резултатите.

Она што треба да знаеме е дека Lucene како основен модел користи комбинација од Vector Space Model (VSM) за IR операции и Boolean model со цел да одлучи колку даден документ е релевантен за корисничкото пребарување. Во основа, идејата позади VSM е дека колку повеќе пати даден збор од Барањето се појавува во даден документ релативно на бројот на појавувања на зборот во целата колекција на документи, толку

[4] https://lucene.apache.org/core/3_6_2/scoring.html

порелевантен е тој документ за Барањето. Boolean моделот се користи за да се изберат документите кои потоа се подредуваат со VSM.

Секако, Lucene овозможува промена на моделот на бодирање (scoring). Во оваа практична имплементација се користи Okapi BM25 моделот кој е опишан во 2.3.2.4. Она што е важно е моделот кој ќе се избере да биде ист при процесот на индексирање и процесот на пребарување.

3 Enterprise Resource Planning (ERP) системи

3.1 ERP дефиниција

ЕРП е софтвер за бизнис менаџмент – вообичаено пакет на интегрирани апликации кои една организација (компанија, фирма, институција) може да ги користи за чување и управување на податоците од секоја позиција на бизнисот.

ЕРП софтверот обезбедува интегриран и во реално време поглед на деловните процеси, користејќи ги заедничките бази на податоци управувани од системот за управување со податоци (ДБМС). ЕРП системот ги следи деловните ресурси – паричните текови, суровините, производствените капацитети – и статусот на деловните обврски: нарачките, набавките, плаќањата, платите, итн. ЕРП апликациите се тие кои овозможуваат заеднички пристап до податоците од различните сектори (производство, набавка, продажба, сметководство, итн) и нивни внес во системот. ЕРП го унапредува текот на информациите помеѓу сите деловните функции, и ги управува врските со надворешните стекхолдери.

Организациите го сметаат ЕРП системот како витална организациска алатка бидејќи ги интегрира различните организациски системи и ги унапредува трансакциите и производството кон помалку грешки. Битна карактеристика која ЕРП системите ги разликува од другите системи е што првите работат на најразличен компјутерски хардвер и мрежни конфигурации, а базата на податоци служи како главен репозиториум на информации.

3.1.1 Историјат

Настанок на терминот ЕРП

Во 1990, Гартнер Груп за првпат го користи акронимот ЕРП како екстензија за МРП (планирање на потреби за материјали), покасно наречено и планирање на ресурси за производство[5] и компјутерски - интегрирано производство. Без замена на овие термини, ЕРП стана претставник на една голема целина која ја рефлектира еволуцијата на апликациите во производството (Sheilds, 2005).

Меѓутоа не сите ЕРП пакети настанале од производното јадро. Различни производители на софтвер (софтверски вендори) почнале со сметководство, одржување

[5] http://www.erp.com/component/content/article/324-erp-archive/4407-erp.html

и човечки ресурси. До средината на 90-тите ЕРП системите ги адресирале сите клучни функции на организациите. Владите и непрофитните организации исто така почнале да користат ЕРП системи (Chang, Gable, Smythe, and Timbrell, 2000).

3.1.2 Експанзија

ЕРП системите доживуваат брза експанзија во 90-тите, дополнително поради познатион ‘problem 2000’ и воведувањето на еврото што доведе до нарушување во работењето на постојните информациски системи. Многу компании ја искористија оваа можност да ги заменат старите системи со ЕРП.

Иницијално ЕРП системите се фокусираа на автоматизација на back office (позадински) функциите кои не ги тангираат директно коминтентите (клиентите) и општо јавноста. Front office функциите, како на пример ЦРМ (модул за менаџирање на односите со коминтентите) кои работат директно со клиентите, или е-бизнис системите како е-комерција, е-влада, е-телеком, и е-финансии или пак СРМ (модул за менаџирање на добавувачите) беа интегрирани во подоцнежните фази од изградбата на ЕРП системите, кога интернетот ја упрости и олесни комуникацијата со надворешните партнери.

Наредната генерација ЕРП системи популарно наречена ЕРП 2, се опишува како веб – базиран софтвер кој им овозможува на вработените и клиентите пристап во реално време до ЕРП системот. ЕРП 2 ја проширува традиционалната оптимизација на ресурси и трансакциското процесирање на ЕРП системите. Наместо само менаџирање со набавките, продажбите, итн. – ЕРП 2 овозможува колаборација меѓу информациските системи меѓу разни организации. Понатаму, флексибилноста му е поголема во однос на претходните ЕРП системи токму поради способноста за интероперабилност со други системи. Алтернативно име за овие системи е EAC (enterprise application suite).

3.1.3 Карактеристики

ЕРП системите типично ги имаат следните карактеристики:

- Интегриран систем кој работи во реално време (или во скоро реално време) без потреба од периодични ажурирања.
- Заедничка база на податоци која ги поддржува сите апликации.
- Инсталација на системот без потреба од разработена апликациска/податочна интеграција од страна на ИТ одделението, со обезбедена имплементација во повеќе мали постепени чекори (Sheilds, 2001).

Функциски области

Еден ЕРП систем ги покрива следните функционални области. Тие се нарекуваат и се групираат во таканаречени ЕРП модули:

- **Финансиско сметководство**: Главна книга, основни средства, обврски кон добавувачите, плаќања, обврски од купувачи, наплата, управување со паричните текови, финансиска консолидација.
- **Планско сметководство**: Буџетирање, трошоци, управување со трошоци, трошоци базирани на активности, итн.
- **Човечки ресурси:** Вработување, обука, плати, бенефиции, менаџирање на разлики, пензионирање, и слично.
- **Производство**: Инженеринг, материјално книговодство, работни налози, операции, капацитет, управување со работни процеси, контрола на квалитет, производствени процеси, производствени текови, управување со животниот тек на продуктите, итн.
- **Процесирање на нарачките**: Нарачки, внесување на нарачки, наредби за исплати на готовински средства, кредитоспособност, ценовници, испорака, анализа на продажбата, провизии за продажба, итн.
- **Управување со набавките**: Планирање на набавките, временско подредување на набавките, конфигурирање на продуктите, продажба и наплата, материјално книговодство, процесирање на барања, магацинско работење (прием, складирање, пакување).
- **Проектен менаџмент**: Проектно планирање, планирање на ресурси, чинење на проектот, разглобување на работата, фактурирање, време и трошоци, мерење на перформаси, управување со активности, и слично.
- **Управување со коминтентите**: продажба и маркетинг, провизии, сервис, контакти со клиентите, call центар поддршка – ЦРМ системите не секогаш се сметале како дел од ЕРП системите туку повеќе како Системи за поддршка на бизнисот (БСС).
- **Податочни сервиси**: Разни самопослужни сервиси наменети за клиентите, добавувачите и вработените.

3.1.4 Компоненти

- Трансакциска база на податоци;
- Менаџерска конзола за управување;

- Business inteligence систем;
- Прилагодливи извештаи;
- Планирање;
- Анализа на продукти;
- Екстерна комуникација преку Веб сервиси;
- Пребарувања;
- Документ менаџмент;
- Пораки/разговори/вики страници;
- Управување со работните процеси.

3.1.5 Предности, корисност и недостатоци

Предности

Основната предност на ЕРП е дека со интеграција на деловните процеси заштедува време и пари. Менаџментот може да носи благовремени одлуки побрзо и со помалку грешки. Податоците се видливи низ целиот деловен систем.

Корисност

- ЕРП може значително да го подобри квалитетот и ефикасноста на бизнисот. Со одржување на непречено работење на организацијата, ЕРП системот овозможува подобри резултати во работењето, во корист на компанијата, како на пример поддршка на коминтентите и производството.
- ЕРП му дава поддршка на врвниот менаџмент обезбедувајќи критични информации на време за носење на важни одлуки.
- ЕРП создава услови за поагилна компанија која полесно се прилагодува на промените (O'Brien, 2011).
- ЕРП ја прави компанијата пофлексибилна и помалку структурно ригидна обезбедувајќи им на деловните единици да работат обединети, унапредувајќи го бизнисот – интерно и екстерно (O'Brien, 2011).

Недостатоци

- Прилагодувањето е проблематично.
- Реинженерингот на процесите заради прилагодување кон ЕРП системот може негативно да влијае на компетитивноста и да го измести фокусот од другите критични активности.

- Цената може да биде поголема одошто цената на помалку интегрирани и посериозни решенија.
- Високата цена за замена на ЕРП систем со друг ги става во неповолна положба корисниците на системот во однос на вендорот.
- Долготрајна обука.
- Хармонизација на разни ЕРП системи може да биде голема работа (особено за големи компании).

Согледувањето на овие недостатоци доведе до развој на нови ЕРП решенија со фокус на неколку критични области: пофлексибилни ЕРП, Веб базирани ЕРП, повеќе компаниски ЕРП, и е-бизнис пакети.

3.2 Пребарување во ERP систем

Пребарувањето на информации низ еден ERP систем е во најдобра рака модуларно ако не функционално, што значи дека можноста за пребарување од страна на корисникот најчесто е имплементирана во рамки на дадена функција или на повисоко на ниво на даден модул. Ефикасноста на пребарувањето на информации зависи од различни информации до кои вработениот има пристап во моментот, односно најчесто вградените пребарувања во ERP системите се дизајнирани да ги вратат сите постоечки резултати согласно филтрирањето кое го поставил корисникот. Постоечките филтри вградени во секое пребарување овозможуваат стеснување и намалување на бројот на потенцијални резултати кои се враќаат на корисникот. Но доколку корисникот нема подетални информации за тоа што го бара тогаш неговото пребарување може да трае подолго.

Не постои совршено пребарување кое ги вклучува сите можни филтри и комбинации на филтри потребни на корисникот. Постојат специјално дизајнирани генератори кои овој проблем го решаваат преку динамичко генерирање на извештаи каде корисникот дефинира филтри во моментот на пребарување и извештаите се дефинираат зависно потребите на корисникот. Сепак овие генератори бараат големо познавање на интерната структура на табелите во базата на податоци како и доста време за да се стигне до релевантниот податок кој е потребен. Истите не се погодни за ад хок пребарувања туку за типови пребарувања кои се почести, пред сè заради релативната спорост во изградбата на овие извештаи.

4 Постоечки методологии за евалуација на IR системи

Во оваа секција е разгледан начинот на кој се врши евалуација на ефективноста и ефикасноста на IR системите. Со цел да се измерат овие параметри на стандарден начин потребна е тест колекција која се состои од три делови (Manning et al., 2009):

1. Колекција на документи;
2. Тест множество на информациски потреби, изразени преку пребарувања;
3. Множество на релевантни проценки на документите, т.е. стандардна бинарна проценка на релевантност или нерелевантност за секој пар пребарување-документ.

Стандарден приод за евалуација на еден IR систем е преку поимите *релевантни* и *нерелевантни* документи. Со почит кон корисничките потреби за информација, секој документ во тест колекцијата добива бинарна класификација како релевантен или нерелевантен. Релевантноста се проценува релативно на информацискатa потреба, а не од зборовите на барањето што значи дека еден документ е релевантен ако ја адресира бараната информациска потреба, а не бидејќи ги содржи сите зборови од барањето. Во оваа фаза може да ја замислиме релевантноста како скала, каде некои документи се многу релевантни додека други помалку. Поедноставен начин е употреба на бинарен начин на одредување на релевантноста како што е прикажан погоре со 0 и 1 соодветно за нерелевантен и релевантен документ.

Сличноста на документ со рангирано пребарување, означена како $S_{q,d}$, укажува на тоа колку блиску содржината на документот укажува на информацискатa потреба изразена со упитот. Со цел да се пресмета сличноста пребарување-документ потребни се најчесто статистички информации за дистрибуцијата на пребарување зборовите – во документите како и во целата колекција. Следејќи ги ознаките и дефинициите на Zobel и Moffat (1998), дефинирани се следните основни статистики за зборови за враќање на документи:

- q, пребарување;
- t, пребарување збор;
- d, документ;
- N_D, бројот на сите документи во колекцијата;
- За секој збор t:
    - $f_{d,t}$, фреквенцијата на t во документ d;
    - N_D, бројот на документи кои го содржат зборот t (без оглед на

фреквенцијата на зборот во секој документ);
    - $f_{q,t}$, фреквенцијата на t во пребарување q;
- За секој документ d:
    - $f_d = |d|$, вкупниот број на појавувања на зборот во d;
- За упитот q:
    - $f_q=|q|$, вкупниот број на појавувања на зборот во q;

Дополнително означени се следните множества:

- D, множество од сите документи во колекцијата;
- D_t, множество од сите документи кои го содржат зборот t;
- T_d, множество од уникатни зборови во документот d;
- T_q, множество на уникатни зборови во упитот и $T_{q,d} = T_q \cap T_d$;

При евалуација на еден IR систем се користат неколку стандардни метрики. Евалуациска метрика се дефинира како множество од мерки кои следат некоја заедничка основна евалуациска методологија. Во традиционалните IR експерименти, најчесто се користи скалата со бинарна релевантност, т.е. вратен документ се вреднува како релевантен или нерелевантен за даденото пребарување.

4.1 Стандардни тест колекции

Постојат два начини за дефинирање на тест колекции: креирање на своја тест колекција или употреба на јавно достапни тест колекции. Во продолжение се прикажани некои достапни тест колекции и евалуациски серии (Manning et al., 2009).

4.1.1 CRANFIELD

Колекцијата Cranfield е првата тест колекција која овозможила прецизни квантитативни мерки за ефективноста на IR системите, но денес е веќе мала за нешто повеќе од елементарни пилот експерименти. Оваа колекција е креирана во доцните 1950-ти и содржи 1398 апстракти од статии во журнали за аеродинамика, множество од 225 пребарувања како и проценка на релевантност за сите парови пребарување-документ.

4.1.2 TREC

Колекцијата Text Retrieval Conference (TREC) е креирана од NIST како резултат од извршувањето на голем број на IR тест евалуациски серии уште од 1992. Во рамките на оваа колекција се користени повеќе тест колекции но помеѓу позначајните се тест колекциите на првите 8 TREC евалуации. Воглавно овие тест колекции се состојат од 6

ЦД дискови кои содржат 1.89 милиони документи и 450 проценки за релевантност за информациски потреби кои се нарекуваат теми и се специфицирани во детални текстуални параграфи.

4.1.3 GOV2

Во последните години, NIST има направено евалуации на поголеми колекции на документи, вклучувајќи ја и 25 милионската GOV2 колекција на веб страници. Од почетокот, NIST тест колекциите на документи биле многу поголеми од било што достапно за истражувачите во тоа време, а GOV2 денес е најголемата Веб колекција која е лесно достапна за истражувачки цели. Сепак, големината на GOV2 колекцијата е сеуште два пати помала од моменталната големина на колекциите индексирани од големите веб пребарувачки компании.

4.2 Евалуација на нерангирани резултати

Двете најчесто користени мерки за мерење на ефективноста на еден IR систем се precision и recall (Manning et al., 2009). Овие најпрво ги дефинираме во наједноставниот случај кога еден IR систем враќа множество на документи за дадено пребарување кои не се рангирани по никаков критериум. Најважните цели на еден IR систем се:

- RECALL – враќање на сите релевантни документи;
- PRECISION – враќање на најрелевантните документи;
- КОМБИНАЦИЈА
    - враќање на што помалку нерелевантни документи,
    - враќање на релевантните документи пред нерелевантните.

Precision (P) е делот на вратени документи кои се релевантни

$$Precision = \frac{\#(\text{вратени релевантни документи})}{\#(\text{вратени документи})}$$

Recall (R) е делот на релевантни документи кои се вратени

$$Recall = \frac{\#(\text{вратени релевантни документи})}{\#(\text{релевантни документи})}$$

Овие поими се уште појасни ако се водиме според следната табела и дополнителната релација.

Табела 4.1. Precision & Recall (Manning et al., 2009)

	Релевантни	**Нерелевантни**

Вратени резултати	True positives (tp)	False positives (fp)
Не вратени	False negatives (fn)	True negatives (tn)

$$P = tp/(tp + fp)$$
$$R = tp/(tp + fn)$$

Предноста од користењето на двете мерки precision и recall е во тоа што во различни услови едната е поважна од другата. Типични веб сурфери сакаат секој резултат на првата страница да е релевантен (висока прецизност) но немаат интерес во преглед на секој документ кој е релевантен. За разлика од нив, различни професионални пребарувачи како на пример разузнавачки аналитичари се многу заинтересирани за добивање на што повисок recall и се подготвени да толерираат и резултати со ниско ниво на прецизност со цел да добијат што повеќе информации. Сепак јасно е дека двете мерки се надополнуваат една со друга. Секогаш може да се добие recall од 1 (со многу ниска прецизност) преку враќање на сите документи за сите пребарувања. И додека recall е неопаѓачка функција според бројот на вратени документи, во еден добар систем прецизноста опаѓа како што бројот на вратени документи се зголемува. Генерално би сакале да добиеме некоја количина на recall со толерирање на само мал процент на false postitives.

Единствена мерка која прави баланс помеѓу овие две мерки се нарекува F-мерка, која претставува тежински хармониски просек (harmonic mean) од precision и recall:

$$F = \frac{1}{\alpha\frac{1}{p} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

каде $\beta^2 = \frac{1-\alpha}{\alpha}$ и $\alpha \in [0,1]$ па $\beta^2 \in [0,\infty]$.

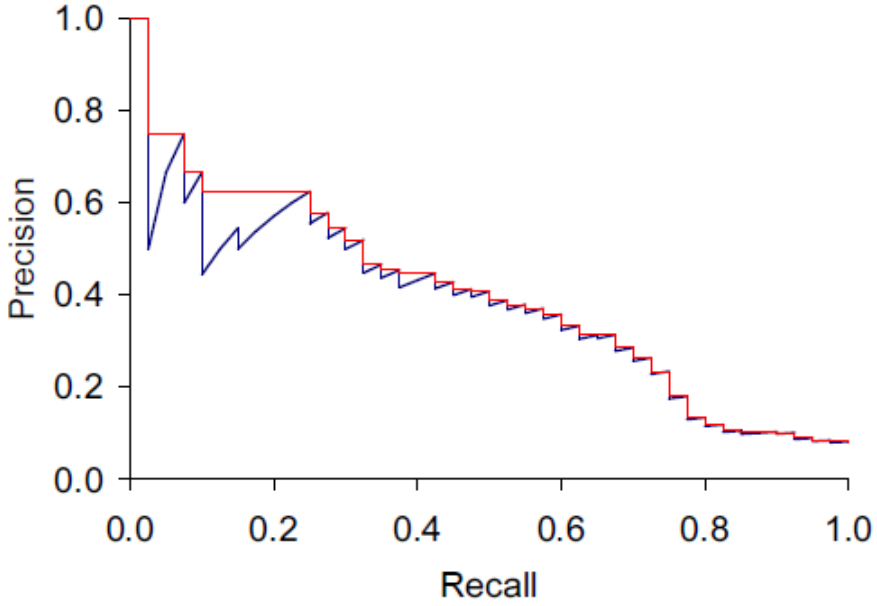
Стандардната балансирана F-мерка постигнува баланс помеѓу precision и recall, што значи дека параметрите ги добиваат следните вредности $\alpha = 1/2$ и $\beta = 1$. Со овие вредности формулата ја добива следната форма:

$$F = \frac{2PR}{P+R}$$

4.3 Евалуација на рангирани резултати

Precision, recall и F-мерка се мерки за ефективност базирани на множества (Manning et al., 2009). Истите се пресметуваат со употреба на неподредени множества на документи. Со цел да се овозможи евалуација на рангирани резултати кои се денес

стандардни за пребарувачките машини, потребно е истите тие мерки да се прошират и надградат. Во контекст на рангирани резултати, соодветни множества на документи се нормално прикажани во првите топ k вратени документи. За секое такво множество, вредностите за precision и recall може да се постават на графикон и да креираат т.н. Precision-recall крива (Слика 4.1.)



Слика 4.1. Пример на precision-recall крива (Manning et al., 2009)

Анализа на кривата precision-recall е многу информативна, но сепак најчесто е пожелно оваа информација да се сведе на еден или неколку броеви. Традиционалниот начин на работа е со помош на просечен precision со интерполација на 11 точки (11-point interpolated average precision). Сепак во последните години се користат и други мерки за евалуација на рангирани резултати.

Перформансите на еден IR систем може да бидат мерени на два различни начина: со *rank cutoff,* каде секоја прецизност или recall се мерени според рангираната позиција откако одреден број на документи се вратени и *overall* каде precision и recall се комбинирани за да креираат единствена вредност за генералните перформанси на IR системот.

Нека со R ја означиме рангираната листа од документи вратена од IR системот како одговор на дадено пребарување, и нека Nrel го претставува вкупниот број на документи кои се оценети како релевантни за тоа пребарување. Понатаму нека rel_r укажува на релевантноста на документ кој има доделен ранг r, така што rel_r=0 доколку документот е нерелевантен и rel_r=1 доколку документот е релевантен. Примери на rank

cutoff и мерки за генералните перформанси користени во IR експерименти ги вклучуваа следните:

- P@r, која ја мери прецизноста до одреден ранг r, и го пресметува делот на вратени документи кои се релевантни на упитот:

$$P@r = \frac{\sum_{i=1}^{r} rel_i}{r}$$

- R@r, која мери recall до одреден ранг r, и го пресметува делот на вратени документи кои се релевантни на упитот:

$$R@r = \frac{\sum_{i=1}^{r} rel_i}{Nrel}$$

- R-Precision (RP), која ја мери прецизноста откако бројот на документи вратени за дадено пребарување е еднаков на бројот на релевантни документи за тоа пребарување:

$$RP = P@Nrel$$

  Доколку бројот на релевантни документи е поголем од бројот на вратени документи, тогаш се претпоставува дека невратените документи не се релевантни за даденото пребарување.

- *Average Precision (AP),* претставува просек од precision пресметана за секое ниво на recall (после секој релевантен документ вратен за упитот):

$$AP = \frac{\sum_{i=1}^{|R|} rel_i \cdot P@i}{Nrel} = \frac{\sum_{i=1}^{|R|} rel_i \cdot P@i}{\sum_{i=1}^{|R|} rel_i} \cdot \frac{\sum_{i=1}^{|R|} rel_i}{Nrel} = \frac{\sum_{i=1}^{|R|} rel_i \cdot P@i}{\sum_{i=1}^{|R|} rel_i} \cdot R@|R|$$

Како последица на погоре наведената дефиниција е тоа што recall-от го ограничува AP, односно еден IR систем чиј recall на долниот крај на рангирањето (R@|R|) е x, може во најдобар случај да пресмета AP од x.

Најстандардна мерка, особено во TREC заедницата, е Mean Average Precision (MAP), која обезбедува единствена мерка за квалитетот на сите нивоа на recall. MAP мерката се покажува особено добра во дискриминација и стабилност. За единствена информациска потреба, MAP претставува просечна вредност од precision добиена од множеството документи од тип k кои постојат после секој релевантен документ кој е вратен како резултат, а понатаму оваа вредност се порамнува во зависност од информациските потреби. Односно, доколку множество на документи за дадена информациска потреба (пребарување) $q_j \in Q$ е $\{d_1, \dots d_{m_j}\}$ и R_{jk} претставува множество на рангирани резултати од топ резултатот додека се дојде до документот d_k тогаш:

$$MAP(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j} Precision(R_{jk})$$

Кога ниту еден релевантен документ не е вратен како резултат, вредноста за precision во погорната равенка е 0. Со употреба на MAP како мерка за ефективност, не постои интерполација и не се прават мерења и се избираат нивоа на recall. MAP вредноста за тест колекцијата претставува аритметичка средина од просечните precision вредности за секое пребарување.

Покрај сите погоре наведени мерки постојат и други релевантни пресметки кои се користат за подетални и посецифични анализи на ефективноста на пребарувачките машини како R-релевантност, ROC крива, Precision-N итн. Сите овие пресметки, зависно од потребата, се показатели на ефективноста на еден IR систем, и по потреба се вклучуваат сите или неколку зависно од значајноста на IR системот и барањата на корисникот.

Дополнителни фактори кои може да бидат земени во предвид при евалуација на ефективноста на пребарувачка машина се следните:

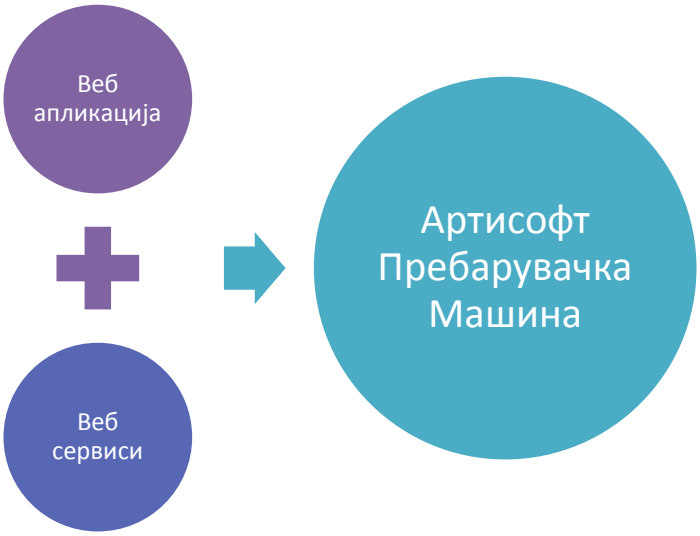
- Брзина на пребарување;
- Зафаќање на ресурси;
- Време на конструкција на индекс од дадена колекција на документи;
- Презентација на вратените резултати.

5 Архитектура на АПМ

Практичната имплементација на пребарувачка машина се нарекува Артисофт Пребарувачка Машина. Основната идеја на оваа практична имплементација е да се развие сервисно и веб - ориентирано софтверско решение кое е интеграбилно, односно може да се користи како пребарувачка машина во рамки на било која апликација и без разлика на документите т.е. содржините низ кои истото ќе пребарува. Се темели на два основни концепта:

- *Plug-n-play концепт на интеграбилност на корисничкиот интерфејс* – овој концепт подразбира дека пребарувачката машина како кориснички интерфејс за пребарување може да се имплементира во било која веб-базирана апликација без разлика на технологијата која се користи за развој на апликацијата, како и независно од технологијата која се користи за развој на самата пребарувачка машина. Ова значи дека сè додека пребарувачката машина е "нахранета" со документите т.е. содржините со кои располага даден систем, пребарувањето низ овие содржини и прикажувањето на резултатите од дадено пребарување може да се извршува од било која веб апликација со едноставно навигирање на претходно дефинирана URL адреса.
- *Концепт на независност од колекција на документи* - овој концепт подразбира независност од видот на документи низ кои сакаме да пребаруваме, како и содржината на истите. Пребарувачката машина се полни, т.е. храни со податоци за овие документи преку јавно достапни веб сервиси кои може да се повикаат од било кое софтверско решение независно од технологијата која се користи за развој на истото. Од овој аспект, Артисофт Пребарувачката Машина претставува сервисно-ориентиран продукт.

Артисофт Пребарувачката Машина претставува интегрирано софтверско решение кое е комбинација од веб апликација и множество на веб сервиси за интерна комуникација во рамки на самата апликација, како и екстерни сервиси за комуникација со надворешни апликации (Слика 5.1.). Токму екстерните сервиси се оние сегменти од решението кои овозможуваат имплементација на Артисофт Пребарувачката Машина на било која веб-базирана апликација, т.е. систем.



Слика 5.1. Основни делови на Артисофт Пребарувачка Машина

Решението поддржува истовремено индексирање и пребарување низ повеќе различни индекси, односно е multi-tenant. Ова значи дека архитектурата на системот поддржува истовремена работа на повеќе клиенти, со што целото решение добива cloud димензија. Секако, возможно е и поставување на решението на дедициран сервер (private cloud) на кој пребарувачката машина работи само за еден клиент.

Во продолжение, најпрво даваме кратко резиме на технологијата која се користи во практичната имплементација, за потоа да поминеме на детален опис на сите компоненти кои се дел од архитектурата на пребарувачката машина. Решението се потпира на Lucene, множество од Јава-базирани библиотеки за индексирање и пребарување на документи. Во поглавието 2.1 даден е преглед на Lucene како пакет, т.е. библиотека за поддршка на IR-системи базирани на текст и детално се опишани сите Lucene објекти кои се користат од страна на компонентите од архитектурата на Артисофт Пребарувачката Машина. Потоа е прикажан интерниот, како и екстерниот дизајн на истата. Во поглавието 5.5 даден е опис на имплементацијата на Артисофт Пребарувачката Машина врз постоечки веб-базиран ЕРП софтвер – АртАИИС. На крајот од ова поглавје, дадена е евалуација на целото решение и презентирани се заклучоците од тестирањето на истот

5.1 Технологија за развој на АПМ

Како што веќе споменавме, Артисофт Пребарувачката Машина е софтверско решение кој претставува комбинација од Веб апликација и множество од Веб сервиси за

комуникација во рамки на истата, како и со надворешни системи врз кои истата би се интегрирала. Развојот на Веб апликацијата и Веб сервисите е изведен параметарски така да поддржува функционалност на различни платформи и технологии.

- *Оперативен систем* – Моменталната имплементација на апликацијата функционира на машина со оперативен систем Windows Server 2008 R2[6], иако истата е поставена и тестирана и на Ubuntu 12.04[7].
- *Веб сервер* – Моменталната имплементација е хостирана на веб сервер IIS7 на оперативен систем Windows Server 2008 R2. Доколку Ubuntu е избран како оперативен систем на кој се поставува апликацијата, како Веб сервер се користи Apache Tomcat 7[8].
- *Апликациски сервер* – Системот моментално работи на апликациски сервер ColdFusion 9[9]. Кодот е напишан во програмскиот јазик ColdFusion. ColdFusion како програмски јазик е речиси идентичен со неколку други програмски јазици кои се функционални на неколку различни платформи, кои за разлика од ColdFusion се бесплатни (open-source). Еден ваков апликациски сервер е и Railo 4.0.[10] Разликите во кодот на апликацијата од ColdFusion до Railo се минимални и се параметарски решени, па апликацијата е тестирана и е функционална и на Railo 4.0. апликацискиот сервер. Со минимални подесувања возможно е миграција на целата имплементација од ColdFusion во Railo околина.
- *База на податоци* – Моменталната имплементација на апликацијата користи PostgreSQL[11] како база на податоци. PostgreSQL е бесплатна релациона база на податоци. Сите делови од кодот кои се користат за комуникација со базата на податоци се консултираат со параметрите дефинирани при поставување и параметризација на имплементацијата каде се одредува од кој тип е базата на податоци која ја користи апликацијата. Кодот поддржува и комуникација со Oracle база на податоци. Со минимални подесувања на параметрите возможно е миграција на апликацијата од PostgreSQL во Oracle околина, како и во обратната насока. Но, миграцијата не подразбира само функционалност на

[6] http://technet.microsoft.com/en-us/evalcenter/dn407368.aspx
[7] http://www.ubuntu.com/download/desktop
[8] http://tomcat.apache.org/download-70.cgi
[9] http://www.adobe.com/products/coldfusion-family.html
[10] http://www.getrailo.com/
[11] http://www.postgresql.org/

кодот кој комуницира со базата на податоци. Потребен е и механизам за миграција на самите податоци кои се веќе зачувани во истата. Овој процес може да биде едноставен, но и доста комплексен, секако во зависност од барањата. Едно покомплексно сценарио би било миграција на податоците од една на друга база на податоци за еден клиент кој ја користи апликацијата, додека во иницијалната база се чуваат податоци за повеќе корисници. Во ваква ситуација се потребни напредни техники за миграција на структурата на базата на податоци и податоците кои се чуваат во истата.

Решението е доволно параметарски дизајнирано, така да овозможува избор на платформа и технологии врз кои истото ќе се постави. Доколку решението се поставува во приватен облак за одреден клиент, останува на клиентот да одлучи комбинација од платформа и технологија на која истото ќе се постави. Ваквиот избор најчесто зависи од моменталната инфраструктура во компанијата на клиентот кај кој би се поставувало решението, како и од цената за поставувањето. Во основа, зависноста во начинот на дефинирање на инфраструктурата постои само меѓу оперативниот систем и веб серверот, додека во поглед на останатите технологии решението е релативно флексибилно. Апликациските сервери (ColdFusion и Railo) и базите на податоци (Oracle и PostgreSQL) се меѓусебно компатибилни.
Веб сервисите кои апликацијата ги користи за комуникација со екстерни системи се независни од платформата и програмскиот јазик во кои се имплементирани ваквите екстерни системи.

5.2 Елементи на архитектурата на АПМ

Пред да ги објасниме клучните компоненти при индексирање и пребарување, ќе дефинираме неколку основни поими:

- *Компанија* – е ентитет кој ги содржи сите информации за даден клиент на кој му е овозможено индексирање и пребарување низ документи. Решението поддржува креирање на повеќе колекции на документи, по една за секој од клиентите, при што сите клиенти ги делат ресурсите на платформата. Решението подржува и можност за миграција и поставување од multi-tenant во single-tenant околина на која перформансите при користење на апликацијата би зависеле само од начинот на кој клиентот ја користи истата, а не и од другите

неактивни клиенти. Секако дека и хардверската платформа влијае на перформансите на системот. Таа може да биде кластерски ориентирана и да поддржи поголеми оптеретувања со примена на балансирање на товарот (load balancing).

- *Корисник* – е ентитет кој ги идентификува корисниците кои пребаруваат во рамки на дадена компанија. Секоја компанија може да има повеќе корисници. Артисофт Пребарувачката Машина овозможува екстерен веб сервис за креирање и мапирање на даден корисник на истата. На овој начин може да се индивидуализира пребарувањето, да се паметат пребарувањата, и да му се предложат сугестии нададениот корисник. Со мапирање на корисниците низ различни системи, се овозможува интеграција на пребарувањето во рамки на овие системи.

Во продолжение ќе ги дефинираме основните IR поими и елементи на Артисофт Пребарувачката Машина.

Документ

Документ е ентитет кој претставува логичка целина од содржина, автор, наслов, опис и сл. кој е од одредена вредност за дадена компанија, но истиот физички не мора да постои. Може да биде збир од записи во база на податоци, извештај добиен од некоја апликација, документ кој физички се чува во некој систем за менаџирање на документи или било кој електронски документ. Постојат два начина на кои Артисофт Пребарувачката Машина "дознава" за постоењето на даден документ и ќе бидат подетално опишани во продолжение.

Документот се води со реден број во база на податоци. Секоја компанија може да чува повеќе документи од различни категории. Податоците кои се водат за истите во базата на податоци ќе бидат подетално опишани во сегментот 5.4.

Секој еден документ кој се индексира мора да се креира како Lucene Document објект, или поточно објект од класата org.apache.lucene.document.Document. На Слика 5.2. е прикажана структурата на еден ваков објект, т.е. множеството од атрибути/полиња (Field) објекти кои се чуваат за даден Document објект. Field објектите се инстанци од класите org.apache.lucene.document.TextField или org.apache.lucene.document.StringField.

Слика 5.2. Структура на Document објект

Полињата содржина, наслов, автор и краток опис се индексираат при градење на индексот. Сите полиња, освен полето содржина се чуваат во рамки на објектот документ, т.е. во самиот индекс. Полето содржина не се чува поради фактот што во таков случај индексот може да бара голема меморија за манипулација и да го забави процесот на пребарување. Сите останати информации се доволни за идентификација на документот и приказ на резултатите при пребарување. URL на документот може да биде мрежно достапна физичка локација на постоечки документ, URL до одреден документ во веб апликација итн. Во продолжение е даден дел од изворниот код (Изворен код 5.1) кој иницијализира објект од типот Document и ги додава Field полињата во рамки на истиот. Компонентата LuceneLoader која се користи во прикажаниот код е објект кој се креира преку компонентата JavaLoader на која како влезни аргументи и се даваат .jar библиотеките на Lucene 4.6.0.

Изворен код 5.1. Иницијализација на објект од типот Document

```
<cfset Document=LuceneLoader.create("org.apache.lucene.document.Document").init()>
<cfset Document.add(LuceneLoader.create("org.apache.lucene.document.TextField").init("content",
doc_content,FieldStoreNo))>
<cfset Document.add(LuceneLoader.create("org.apache.lucene.document.StringField").init("id",
document_id.toString(),FieldStoreYes))>
<cfset Document.add(LuceneLoader.create("org.apache.lucene.document.StringField").init("url",
document_link.toString(),FieldStoreYes))>
<cfset
Document.add(LuceneLoader.create("org.apache.lucene.document.StringField").init("document_name",
document_name,FieldStoreYes))>
<cfset Document.add(LuceneLoader.create("org.apache.lucene.document.StringField").init("author",
author,FieldStoreYes))>
<cfset
Document.add(LuceneLoader.create("org.apache.lucene.document.StringField").init("document_description
",mid(doc_content,1,100),FieldStoreYes))>
```

Стоп зборови

Стоп зборови е множество од зборови за дадена компанија кои треба да се игнорираат при индексирање и пребарување. Овој ентитет е опишан и во интерниот дизајн на апликацијата. Секоја компанија си го параметризира ова множество на зборови. Стоп зборовите се најчесто сврзници, предлози и слични зборови кои не се многу значајни при пребарување. Од аспект на индексирање, ваквите зборови значително придонесуваат во намалување на големината на индексот.

Стоп карактери

Стоп карактери е множество од специјални карактери кои се користат за поделба на даден текст на зборови. Иницијален стоп карактер е празното место. Дополнително стоп карактери може да бидат карактери како “,./\’;][!@#$%^&*()_+=” итн.

Мора да напоменеме дека колку и да изгледа едноставно, мора да се посвети внимание при дефинирање на ваквите карактери, бидејќи истите може значајно да влијаат на пребарувањето. Пример за вакво сценарио е користење на “.” како стоп карактер. На крајот на реченица таа е стоп карактер меѓутоа при индексирање на датумот 22.03.2013, зборовите кои ќе се индексираат се “22”, “03” и “2013”. Ова можеби е точно, но важно е да бидеме свесни како дефинирањето на ваквите карактери ќе влијае на пребарувањето и подредувањето на резултатите. Стоп карактерот како ентитет е

опишан и во интерниот дизајн на апликацијата во сегментот 5.4.

Парсер

Парсерот е ентитет кој во интерниот дизајн на апликацијата се нарекува Тип на парсер. Всушност самата операција на парсирање се извршува преку Lucene објектот Analyzer, поточно објект од класата org.apache.lucene.analysis.standard.StandardAnalyzer. Тип на парсер како ентитет се користи заради дефинирање на различни комбинации на стоп зборови и стоп карактери при индексирање на различни видови документи. При едноставно пребарување се користи стандарден парсер.

Со можноста за дефинирање на повеќе типови на парсери оставаме простор за надградба на системот и користење на различни типови на парсери и при самото пребарување, при што ќе се дефинира типот на документи низ кои се пребарува. На овој начин би можеле да имплементираме напредни пребарувачки техники. Во продолжение е даден дел од код кој користи метод од даден сервис кој враќа листа од стоп зборови и потоа креира Lucene Analyzer објект на кој како параметар му се предава низата од стоп зборови.

Изворен код 5.2. Lucene Analyzer објект

```
<cfinvoke component="search_engine" method="select_ow" returnvariable="qSw">
        <cfinvokeargument name="id_company" value="#id_company#">
        <cfinvokeargument name="id_parser" value="#id_parser#">
</cfinvoke>
<cfset StopWords = LuceneLoader.create("org.apache.lucene.analysis.util.CharArraySet").init
(LuceneVersion, qSw.RecordCount, true)>
<cfloop list="#qSw.word_naziv#" delimiters="," index="i">
        <cfset sw=i.toCharArray()>
        <cfset StopWords.add(sw)>
</cfloop>
<cfset Parser= LuceneLoader.create
("org.apache.lucene.analysis.standard.StandardAnalyzer").init(LuceneVersion, StopWords)>
```

Индекс

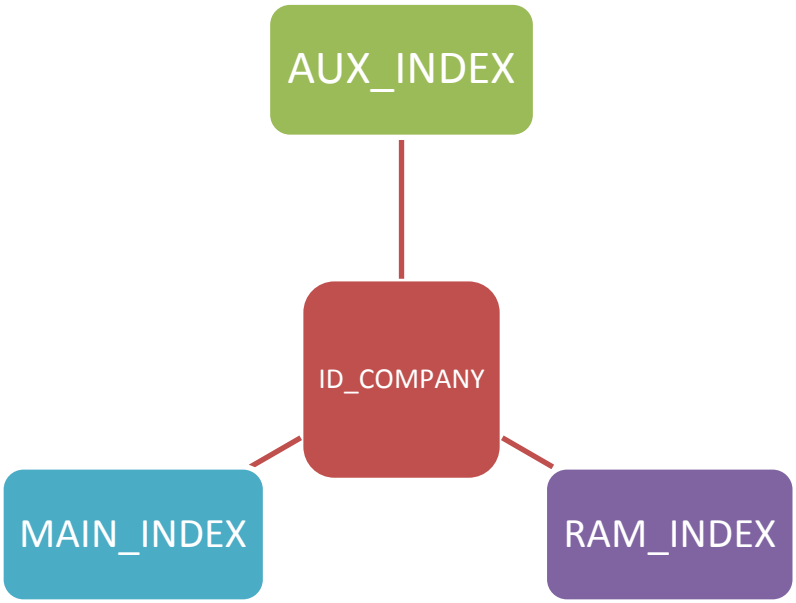
Веќе дадовме дефиниција на инвертиран индекс како поим. Целата комплексност на форматот на зачувување, заедно со сите информации и врски меѓу објектите кои се креираат при градење на индексот, е задача која ја извршува Lucene. Она што треба да се дефинира при генерирање на индексот е неговата физичка локација, односно директориумот во кој истиот ќе се чува.

Сепак, не е доволно да се дефинира само еден директориум, и тоа од повеќе причини. Една од причините е поддршката за истовремена работа на повеќе клиенти. Иако во основа за секој документ може да се води и атрибут компанија на која истиот припаѓа и физички да постои еден директориум во кој се чува индексот, сепак подобро решение е секоја од компаниите да има засебен директориум. Одлуката за ваквиот начин на дефинирање на индексот е заради:

- *Брзина на индексирање/пребарување* – Доколку се чува еден индекс за сите компании, тој индекс ќе биде поголем, потежок за манипулација, па соодветно и процесот на градењето на индексот и пребарувањето ќе биде поспоро. Перформансите значително ќе се намалат, дури и за клиенти кои реално немаат голем број на документи и зборови, за сметка на оние кои имаат голем број на документи и разновидни зборови.
- *Потреба од миграција на податоци* – Доколку физички индексот се чува во ист директориум за сите клиенти, тогаш процесот на миграција на податоци за дадена компанија би бил доста комплексен и би барал огромни временски ресурси, па може да се каже дека е и практично невозможен. Чувањето на засебни индекси за секој од клиентите оваа операција ја поедноставува со "copy-paste" на директориумот во кој се чува индексот од една на друга физичка локација.

И покрај чувањето на засебни индекси за секој клиент, т.е. компанија, сепак индексот за една компанија повторно не е само еден директориум. Индексирањето на нови документи и нивната содржина е динамичен процес и операција која треба да биде возможна во било кое време. Сепак, додавањето на нов документ во индексот е операција која истиот го заклучува и тој не може да се користи за пребарување. Ова наметнува потреба за дефинирање на дополнителни датотеки, како копии на индексот.

За Lucene индексот е збир од датотеки кои се запишуваат во еден директориум и се меѓусебно зависни. За да ја елиминираме оваа комплексност индексот го нарекуваме директориум и она што е важно е во секој момент да ја знаеме локацијата на која треба да запишуваме и онаа од која треба да читаме при пребарување. Според тоа, индексот, т.е. директориумот за секоја компанија е со назив *id_company* – единствен идентификатор за секоја компанија, чија структура е прикажана на Слика 5.3.



Слика 5.3. Структура на индекс на една компанија

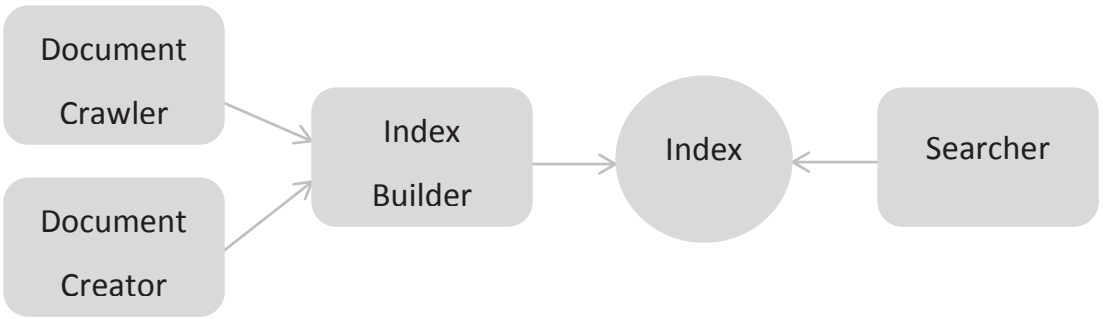
Индексот на една компанија е структура од повеќе директориуми:

- AUX_INDEX – е помошен индекс во кој се чуваат сите документи и зборови за компанијата. Овој индекс е цело време ажурен. При додавање на нов документ се користи овој директориум. Овој индекс не се користи при процесот на пребарување.
- MAIN_INDEX – е главен индекс кој се користи за пребарување на документи. Овој индекс не е цело време ажурен, токму поради фактот што треба цело време да биде достапен за запишување. Податоците од помошниот индекс периодично се префрлуваат во овој директориум и тоа со посебна операција за префрлување која се изведува во специфично време од денот во кое има најмалку активност од страна на корисниците. Фреквенцијата на префрлување на овие податоци од помошниот во главниот индекс може да се дефинира за секоја компанија индивидулано, во зависност од фреквенцијата на промена и додавање на документи во системот.
- RAM_INDEX – е копија на главниот индекс која се креира како објект при иницијализација, односно стартување на апликацискиот сервер. Референца кон овој објект се чува сè додека е функционален апликацискиот сервер и истата се користи за активно пребарување. Причината за постоењето на овој вид на индекс е брзината на пребарување која е поголема отколку при вчитување на

индексот од секундарната меморија (хард диск) во моментот на пребарување. При користење на оваа техника за забрзување на пребарувањето важно е да се води сметка за големината на индексот на компанијата и големината на главната меморија – RAM на машината.

5.3 Компоненти на АПМ

Елементите од архитектурата на системот кои досега ги опишавме играат главна улога во основните операции на пребарувачката машина. На Слика 5.4 е прикажана меѓусебната интеракција на основните компоненти во системот, а во продолжение детално ќе бидат опишани истите.



Слика 5.4. Основни компоненти на пребарувачката машина

Екстерен сервис за пронаоѓање/креирање на документи

Компонентите кои комуницираат, односно ја повикуваат компонентата *Index Builder* (documentCrawler, documentCreator) се всушност екстерните сервиси преку кои било кој систем кој е регистриран како компанија која ќе ја користи пребарувачката машина ја "храни" истата со документи. Задачата на овие компоненти е да дознаат за постоењето на еден или пак група на документи и истите да ги проследат до компонентата која ќе ги индексира. Постојат два начина на работа:

- *Document Crawler* – е компонента која комуницира со надворешни системи заради пронаоѓање на документи кои на одреден начин се означени дека треба да се индексираат. Оваа компонента се повикува преку методот *documentCrawler* од јавно достапен веб сервис и како аргументи прима корисничко име и лозинка заради автентификација на истиот и идентификација на компанијата која индексира, како и патека на мрежно достапна локација,

заедно со корисничко име и лозинка за најава на оваа мрежно достапна локација. Алгоритмот е креиран да ги измине рекурзивно сите директориуми на дадената локација и да ги собере сите пронајдени документи. Секој од овие документи се процесира и според форматот на алгоритмот се повикува компонента која ја превзема содржината на истиот во текст формат, како и други податоци од типот на наслов, автор и сл. Методот *documentCrawler* како аргумент прима и URL адреса на која потоа ќе може да се пристапи до документот. По пронаоѓање на сите документи кои системот треба да ги испроцесира се повикува *Index Builder* компонентата која ги индексира пронајдените документи. Во продолжение е дадена дефиницијата на методот *documentCrawler*.

Изворен код 5.3. Дефиницијата на методот documentCrawler

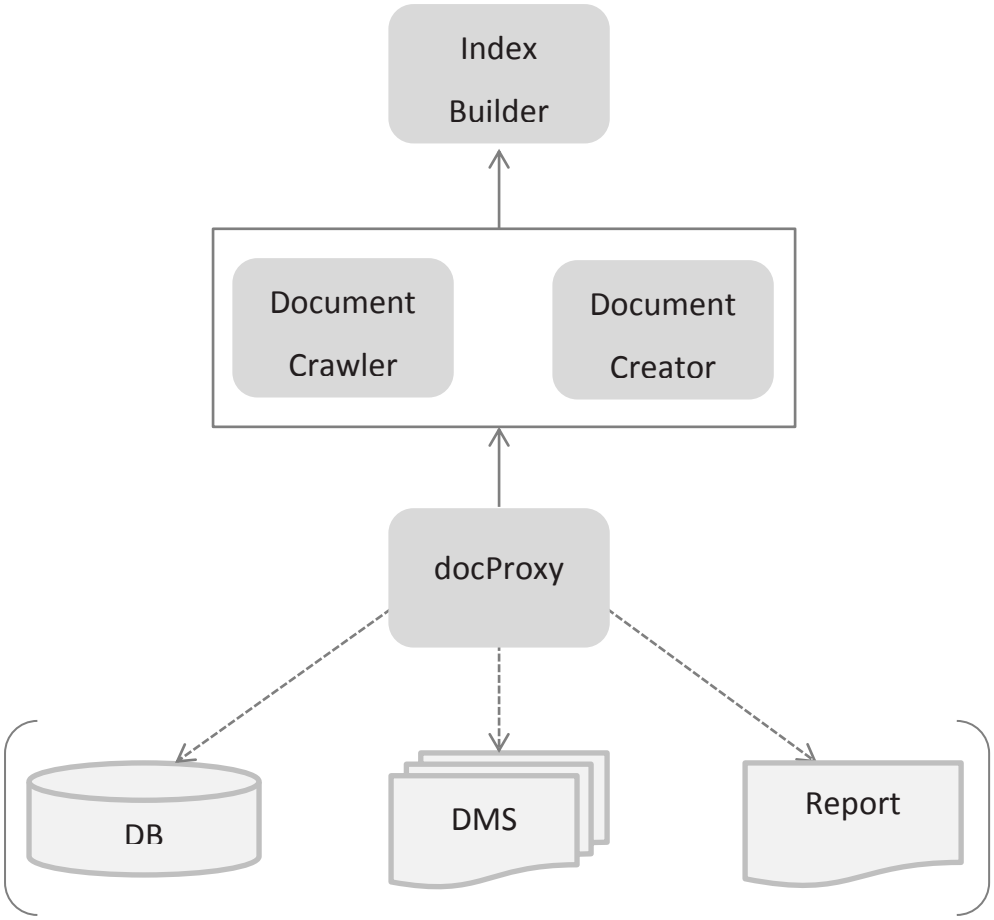
```
<cffunction name="documentCrawler" access="remote" output="Yes" returntype="array" >
        <cfargument name="username" type="string" required="Yes">
        <cfargument name="password" type="string" required="Yes">
        <cfargument name="url" type="string" required="Yes">
        <cfargument name="path" type="string" required="Yes">
        <cfargument name="path_username" type="string" required="Yes">
        <cfargument name="path_password" type="string" required="Yes">
        <cfinclude template="search_engine/polni_index/main.cfm">
        <cfreturn results>
</cffunction>
```

- *Document Creator* - е компонента која директно може да се повика од било кој надворешен систем во вид на веб сервис. Оваа компонента се повикува преку методот *documentCreator* од јавно достапен веб сервис и како аргументи прима корисничко име и лозинка заради автентификација на сервисот и идентификација на компанијата која индексира, како и податоци како наслов на документ, URL адреса и еден од аргументите содржина на документ во текст форма или самиот документ во бинарна форма. Оваа компонента се користи за обработка на документот во моментот на негово креирање при што сервисот ја експортира содржината на документот, или пак ја добива во форма на текст како влезен аргумент. По процесирање на документот се повикува *Index Builder* компонентата која го индексира истиот. Во продолжение е дадена дефиницијата на методот *documentCreator*.

Изворен код 5.4. Дефиницијата на методот documentCreator

```
<cffunction name="documentCreator" access="remote" output="Yes" returntype="string" >
        <cfargument name="username" type="string" required="Yes">
        <cfargument name="password" type="string" required="Yes">
        <cfargument name="link" type="string" required="Yes">
        <cfargument name="document_name" type="string" required="No" default="">
        <cfargument name="document_content" type="string" required="No" default="">
        <cfargument name="documentToUpload" type="binary" required="No">
        <cfinclude template="search_engine/polni_index/main.cfm">
        <cfreturn results>
</cffunction>
```

Паралела на овие компоненти е Веб crawler, робот компонента која го изминува веб-от и ги индексира новите документи и веќе ја споменавме во **Error! Reference source not ound.**. На Слика 5.5. е прикажана интеракцијата на овие компоненти со останатите делови од Артисофт Пребарувачката Машина, како и со екстерните системи.



Слика 5.5. Интеракција на documentCrawler и documentCreator со останатите компоненти

Веќе споменавме дека компонентите documentCrawler и documentCreator ја повикуваат Index Builder компонентата. Но, за да се повикаат ваквите компоненти документите мора претходно да постојат и да се повлечат од некои надворешни системи, како база на податоци, ДМС системи или било каков друг податочен систем. Во случај на креирање на нови документи, во некој од стандардните формати (pdf, excel, word, txt итн.) или пак во вид на текст се повикува documentCreator компонентата.

Во случајот кога треба да се обработат веќе постоечки документи има потреба од дополнителни подесувања: документите се сместуваат на мрежно достапна локација и се процесираат или пак се пишуваат посебни програми кои ќе обработат одредена колекција на документи или записи од база на податоци и сл. Овој код го нарекуваме docProxy и е прашање на имплементација на секој систем кој ќе ја користи пребарувачката машина. Во основа, docProxy не треба да биде комплексен код, баш напротив, релативно лесно за имплементација за оние кои го познаваат системот кој се имплементира. Во сегментот 5.5 е опишана конкретна имплементација на една ваква компонента за веќе постоечки систем.

Index Builder

Оваа компонента игра една од клучните улоги во целиот систем и се користи за индексирање на еден или пак група на документи. Се повикува од една од компонентите documentCrawler или documentCreator кои како резултат даваат множество на документи.

Всушност, задачата на оваа компонента е да запише еден или множество на документи во т.н. помошен индекс на компанијата на која ѝ припаѓаат документите, односно во директориумот *ID_COMPANY/AUX_INDEX*. За таа цел се креира Lucene објект од типот FSDirectory (org.apache.lucene.store.FSDirectory) . Во продолжение е даден кодот кое ја извршува оваа операција.

Изворен код 5.5. Објект FSDirectory

```
<cfset IndexDirectory=LuceneLoader.create("org.apache.lucene.store.FSDirectory").
open(LuceneLoader.create("java.io.File").init("#Application.index_folder#/#id_company#/aux_index"))>
```

При иницијализација на повеќе Lucene објекти, креаторот на објектите како влезен аргумент ја бара верзијата на Lucene, објект кој се добива со следниот код:

Изворен код 5.6.Lucene верзија

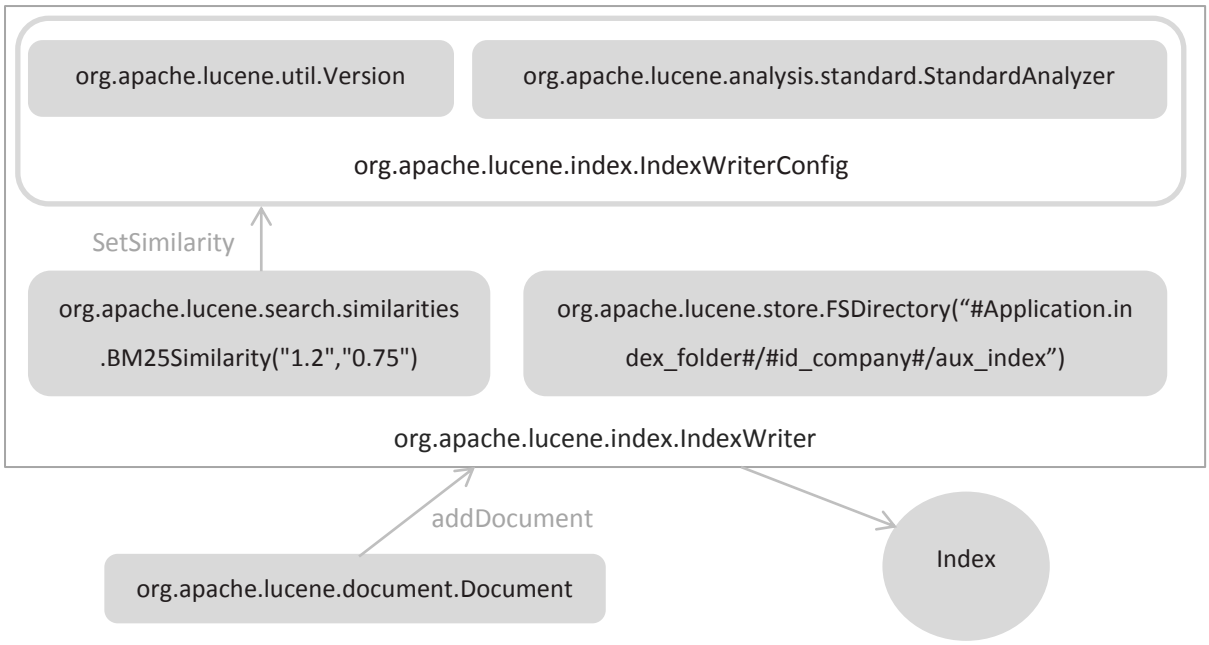
```
<cfset LuceneVersion = LuceneLoader.create("org.apache.lucene.util.Version").LUCENE_CURRENT>
```

Споменавме на кој начин се креира објектот од тип Analyzer, т.е. парсерот кој се користи при пребарување и индексирање на документи. Овој објект се користи како влезен аргумент при креирање на објект од типот IndexWriterConfig (класа org.apache.lucene.index.IndexWriterConfig). IndexWriterConfig објектот се користи за подесување на парсерот и моделот за пресметување на scoring на документот, за потоа да се користи како влезен аргумент при иницијализација на IndexWriter објектот (класа org.apache.lucene.index.IndexWriter). Во продолжение е даден кодот кој креира IndexWriterConfig објект, го поставува моделот на scoring Okapi BM25 моделот и креира IndexWriter објект кој ги индексира документите. Откако ќе се креира Document објект (сегмент 5.2) истиот се додава во индексот преку addDocument методот на IndexWriter објектот. На крај се уништуваат референците кон сите Lucene објекти.

Изворен код 5.7.Okapi BM25 модел во Lucene

```
<cfset IndexWriterConfigurator =
LuceneLoader.create("org.apache.lucene.index.IndexWriterConfig").init(LuceneVersion,Parser)>
<cfset BM25Similarity=LuceneLoader.create
("org.apache.lucene.search.similarities.BM25Similarity").init("1.2","0.75")>
<cfset IndexWriterConfigurator.setSimilarity(BM25Similarity)>
<cfset IndexWriter =
LuceneLoader.create("org.apache.lucene.index.IndexWriter").init(IndexDirectory,IndexWriterConfigurator)>
<cfset IndexWriter.addDocument(Document)>
```

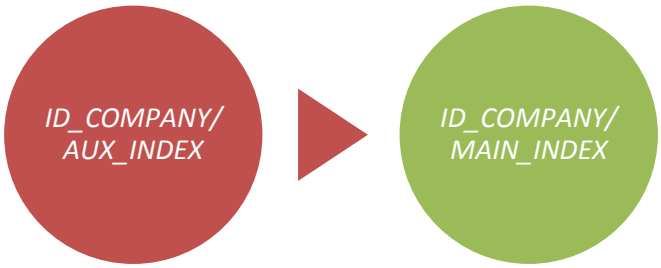
На Слика 5.6. е прикажана интеракцијата меѓу различните Lucene објекти кои се користат при извршување на Index Builder компонентата.



Слика 5.6. Интеракција меѓу Lucene објекти во Index Builder компонетата

Ажурирање на главен индекс

При индексирање на документите истите се додаваат во директориумот *ID_COMPANY/AUX_INDEX*. Повремено, потребно е копирање на содржината од овој директориум во директориумот *ID_COMPANY/MAIN_INDEX* (Слика 5.7). Ова претставува едноставна copy-paste операција. Единствено што треба да се внимава е времето во денот кога ќе се изврши оваа операција, односно време кога корисниците на апликацијата за дадена компанија се најмалку активни. Ваквите анализи се возможни од логовите на корисници и записите за пребарувањата кои тие ги извршуваат.



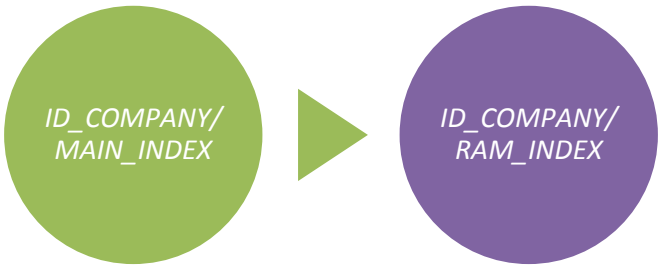
Слика 5.7. Освежување на главен индекс

Освен избор на време во денот кога ќе се извршува операција на ажурирање на индексот, важен параметар е и фреквенцијата на повторување. За таа цел, овозможено е самите компании да дефинираат колку често ќе се ажурира индексот или пак максимален број на новододадени документи во помошниот индекс, по што неговата содржина треба да се префрли во главниот индекс.

Оваа операција се решава со воведување на техники како scheduler-и кои се активираат во одредено време од денот и според одредени параметри одлучуваат дали и кои директориуми треба да се ископираат.

Префрлување на главен индекс во RAM меморија

Една интересна, но незадолжителна операција е префрлувањето на главниот индекс во RAM меморија (Слика 5.8).



Слика 5.8. Префрлување на главен индекс во RAM меморија

Оваа операција значи креирање на серверска променлива која е активна сè додека работи апликацискиот сервер и истата референцира кон објект од класа RAMDirectory (org.apache.lucene.store.RAMDirectory) кој е копија на директориумот *ID_COMPANY/MAIN_INDEX*. Префрлувањето на главниот индекс во RAM е битна операција од аспект на подобрување на перформансите при пребарување. Имено, пребарувањето е побрзо доколку индексот се наоѓа во RAM. Сепак, потребно е да се внимава при користењето на оваа техника. Потребно е вчитување на n индекси во RAM, каде n е бројот на компании. Потребно е да се внимава и на големината на ваквите индекси.Префрлувањето на индексот во RAM се извршува во моментот на иницијализација, т.е. стартување на апликацискиот сервер. Во администраторот на истиот се поставува патека до Server.cfc компонента во која постои метод *onServerStart* кој се извршува при старт/рестарт на апликацискиот сервер. Во продолжение е даден извадок од кодот на овој метод.

Изворен код 5.8.Префрлувањето на индексот во RAM

```
<cfcomponent displayname="Server" output="true">
    <cffunction name="onServerStart" access="remote" returntype="boolean" output="true" hint="Fires
when the server is started.">
        <cfquery name="qAllCompanies" datasource="search_engine">….</cfquery>
            <cfloop query="qCompanies">
                …
                <cfset "SERVER.CompanyRAMDirectory_#id_company#"=
LuceneLoader.create("org.apache.lucene.store.RAMDirectory").init(LuceneLoader.create("org.apache.lucene
.store.FSDirectory").open(LuceneLoader.create("java.io.File").init("#Application.index_folder#/#id_compan
y#/main_index")),LuceneLoader.create("org.apache.lucene.store.IOContext").DEFAULT)>
                …
            </cfloop>
            <cfreturn true />
    </cffunction>
</cfcomponent>
```

Освен при стартување на апликацискиот сервер, главниот индекс се префрлува во RAM и при ажурирање на главниот индекс.

Searcher

Searcher (Пребарувач) е компонента која се користи за пребарување низ документите. Оваа компонента како влезни аргументи ги прима идентификаторот на корисникот кој пребарува, со што се идентификува и компанијата, како и пребарувачките зборови.

Задачата на оваа компонента е да пребарува низ директориумот кој се чува во RAM меморија, или пак низ главниот индекс, доколку не постои индекс во RAM меморија. Во следниот код се креира референца кон индексот:

Изворен код 5.9.Пребарување низ директориум во RAM меморија и низ главниот индекс

```
<cfif not isdefined("SERVsearch_queryER.CompanyRAMDirectory_#id_company#")>
    <cfset CompanyRAMDirectory=LuceneLoader.create ("org.apache.lucene.store.RAMDirectory")
.init(LuceneLoader.create("org.apache.lucene.store.FSDirectory").open(LuceneLoader.create("java.io.File").
init("#Application.index_folder#/#id_company#/main_index")),LuceneLoader.create("org.apache.lucene.sto
re.IOContext").READ)>
    <cfelse>
        <cfset CompanyRAMDirectory=evaluate("SERVER.CompanyRAMDirectory_#id_company#")>
</cfif>
```

На сличен начин како и при индексирање на документи, се креира објект од тип Analyzer, односно парсер кој пребарувањето ќе го раздели на зборови преку објект од

типот ПребарувањеParser (org.apache.lucene.пребарувањеparser.classic.ПребарувањеParser). При креирање на овој објект се дефинира и по која од колоните кои се индексирани ќе се пребарува. Во овој случај тоа е колоната “content”. Следниот код го креира ваквиот објект и го парсира упитот внесен од корисникот - search_query.

Изворен код 5.10.Креирање на објект ПребарувањеParser

```
<cfset QueryParser = LuceneLoader.create("org.apache.lucene.queryparser.classic.QueryParser").
init(LuceneVersion,"content",Parser)>
<cfset Query=QueryParser.parse(search_query)>
```

Потоа се креира објект од тип DirectoryReader (org.apache.lucene.index.DirectoryReader) кој како влезен аргумент го прима претходно креираниот објект кој покажува кон индексот. Овој објект потоа се користи како влезен аргумент при креирање на објектот IndexSearcher (класа org.apache.lucene.search.IndexSearcher). Објектот IndexSearcher има метод *setSimilarity* кој како аргумент прима референца кон Similarity објект преку кој се дефинира моделот според кој ќе се сортираат резултатите. Моделот при индексирање треба да биде ист со моделот кој се користи при пребарување. Во двата случаи, како што кажавме и претходно овој модел е Okapi BM25 со тежини 1.2 и 0.75. Во продолжение е даден кодот кој ги извршува опишаните операции:

Изворен код 5.11.Поставување на модел за пребарување

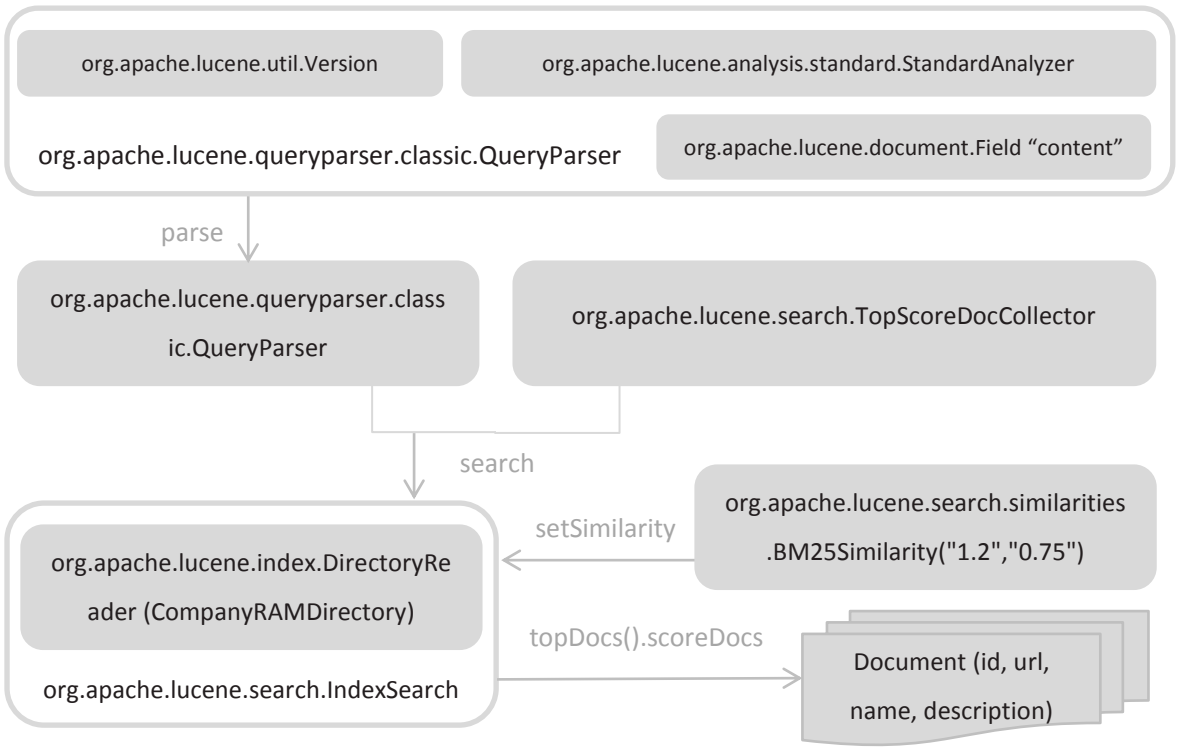
```
<cfset IndexReader =
LuceneLoader.create("org.apache.lucene.index.DirectoryReader").open(CompanyRAMDirectory)>
<cfset IndexSearcher = LuceneLoader.create("org.apache.lucene.search.IndexSearcher").init(IndexReader)>
<cfset BM25Similarity=LuceneLoader.create("org.apache.lucene.search.similarities.BM25Similarity").
init("1.2","0.75")>
<cfset indexSearcher.setSimilarity(BM25Similarity)>
```

Објектот IndexSearcher има метод *search* кој всушност го извршува пребарувањето. Овој метод како влезни аргументи ги прима Query објектот и објект од типот TopScoreDocCollector кој се користи за подредување на резултатите според избраниот модел. TopScoreDocCollector има метод topDocs кој ги враќа документите во низа. Со следниот код се извршува пребарувањето преку IndexSearcher и приказот на резултатите во формат *document_name, url, document_description*.

Изворен код 5.12.Поставување на модел за пребарување

```
<cfset TopScoreDocCollector =
LuceneLoader.create("org.apache.lucene.search.TopScoreDocCollector").create(1000, true)>
<cfset IndexSearcher.search(Query,TopScoreDocCollector)>
<cfset hits= ArrayNew(1)>
<cfset hits=TopScoreDocCollector.topDocs().scoreDocs>
<cfloop from="1" to="#ArrayLen(hits)#" index="i">
        <cfset docId=hits[i].doc>
        <cfset d=IndexSearcher.doc(docId)>
        #d.get('document_name')#, #d.get('url')#, #d.get('document_description')#
</cfloop>
```

На Слика 5.9. е прикажана интеракцијата меѓу Lucene објектите при извршување на Searcher компонентата.



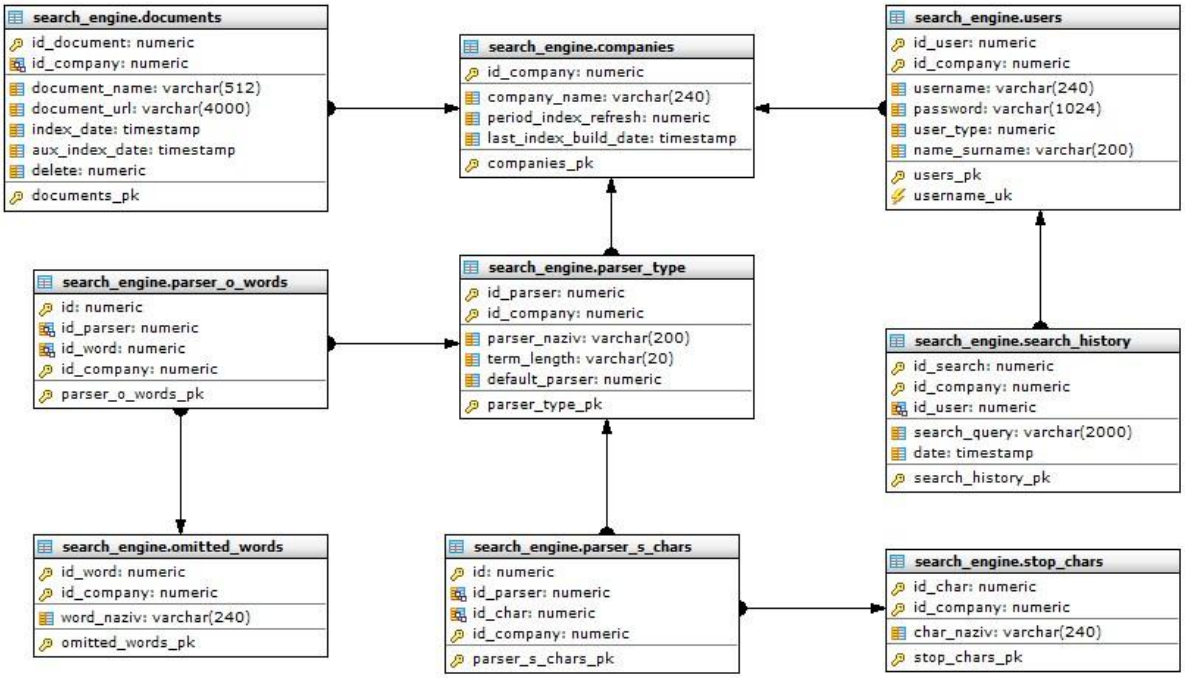
Слика 5.9. Интеракција меѓу Lucene објекти при извршување на Searcher компонентата

5.4 Дизајн на АПМ

5.4.1 Интерен дизајн

Во овој сегмент накратко ќе ги резимираме ентитетите кои се дел од релационата база на податоци на Артисофт Пребарувачката Машина. Од Слика 5.10, на која е прикажан

ЕР дијаграмот на пребарувачката машина, евидентно е дека нема голем број на табели и дека целиот дизајн е релативно едноставен. Во продолжение накратко е опишано за што се користи секој од ентитетите во овој дијаграм.



Слика 5.10. ЕР дијаграм на Пребарувачка Машина

- COMPANIES – е ентитетот *Компании*. Овој ентитет се користи за чување на податоци за сите клиенти кои ја користат пребарувачката машина, како идентификатор, назив на компанија, последен датум на префрлување од помошен во главен индекс, период на освежување на индекс и сл.
- USERS – е ентитетот *Корисници*. Овој ентитет се користи за водење на евиденција за сите корисници во рамки на една компанија и податоци за истите како идентификатор на корисник, идентификатор на компанија, корисничко име, лозинка, тип на корисник и име и презиме.
- DOCUMENTS – е ентитетот *Документи*. Овој ентитет се користи за чување на податоци за документите кои се индексираат. За секоја компанија може да се чуваат повеќе документи за кои се водат следните атрибути: идентификатор на документот, идентификатор на компанијата, назив на документот, URL адреса, датум на внес во помошен индекс, датум на внес во главен индекс, индикатор дали документот треба да биде избришан од индексот.
- SEARCH_HISTORY – е ентитетот *Историјат на пребарување*. Самиот назив на

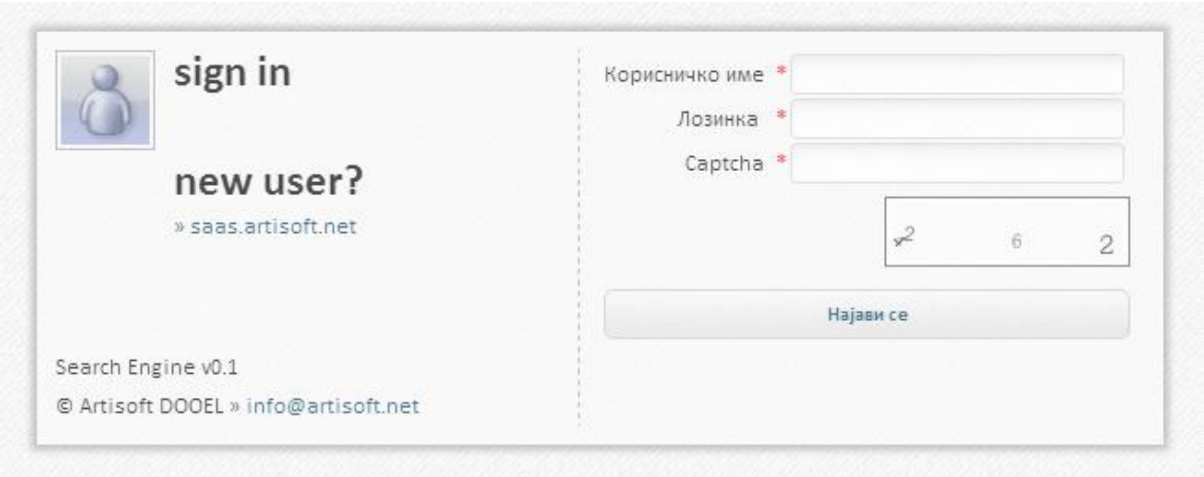
овој ентитет објаснува каков вид на податоци се чуваат во оваа табела. Податоците кои се водат се: идентификатор на пребарувањето, идентификатор на компанијата, идентификатор на корисникот кој го извршил пребарувањето, датум на пребарување и клучни зборови во вид на стринг.

- STOP_CHARS – е ентитетот Стоп карактери кој веќе го објаснивме во секција 5.2. Во оваа табела се чуваат податоци за стоп карактерите во дадена компанија преку записи кои ги содржат следните атрибути: идентификатор на стоп карактерот, стоп карактерот, идентификатор на компанијата.
- OMITTED_WORDS – е ентитетот *Исклучни (стоп) зборови* кој веќе го објаснивме во секција 5.2. Во оваа табела се чуваат податоци за стоп зборовите во дадена компанија преку записи кои ги содржат следните атрибути: идентификатор на стоп зборот, самиот стоп збор, идентификатор на компанијата.
- PARSER_TYPE е ентитетот *Тип на парсер*. Овој ентитет ја дефинира структурата на табелата во која се чуваат податоци за типовите на парсери во дадена компанија. Атрибутите кои се водат за секој тип на парсер се: идентификатор на парсерот, идентификатор на компанијата, назив на парсерот, индикатор дали типот на парсер е default парсер кој се користи доколку не е специфицирано кој тип на парсер треба да се користи при индексирање/пребарување на податоци.
- PARSER_S_CHARS – е ентитет во кој се чува врската помеѓу ентитетот *Тип на парсер* и ентитетот *Стоп карактери*. Оваа врска ги означува сите стоп карактери кои важат за секој од постоечките типови на парсери.
- PARSER_O_WORDS - е ентитет во кој се чува врската помеѓу ентитетот *Тип на парсер* и ентитетот *Исклучни зборови*. Оваа врска ги означува сите исклучни зборови кои важат за секој од постоечките типови на парсери.

Документите се единствениот тип на податок кој се чува и во релационата база на податоци и во самиот индекс, иако податоците кои се чуваат се разликуваат во одреден дел и се взаемно исклучливи. Податоците кои се повторуваат и во Lucene-базираниот индекс и во релационата база на податоци се чуваат заради контрола. Типот на парсер е ентитет чии податоци за конфигурација се чуваат во релационата база на податоци, додека имплементацијата на парсерот е дел од Lucene класите и методите.

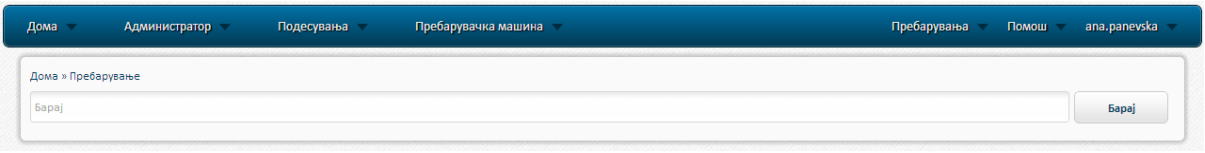
5.4.2 Екстерен дизајн

По логирање во апликацијата пополнувајќи ги податоците на Слика 5.11. се отвара екранот прикажан на Слика 4.12.



Слика 5.11. Екран за најавување во Артисофт Пребарувачката Машина

Корисникот кој е најавен на Слика 5.12. е од тип администратор, т.е. вработен во Артисофт и има овластувања за сите функции во апликацијата.

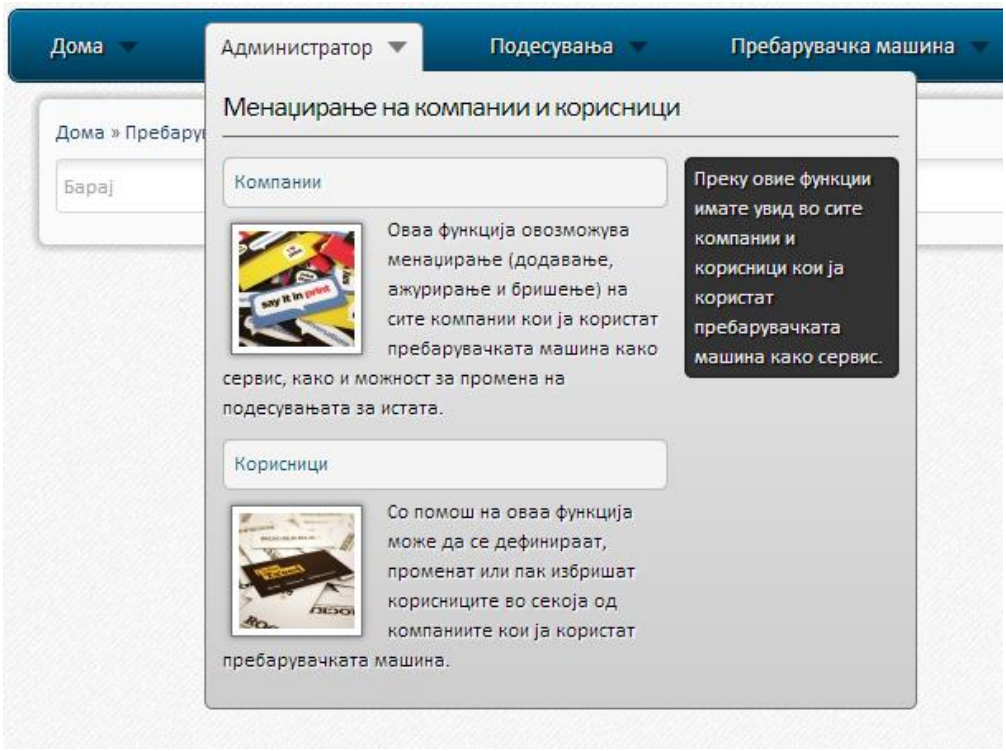


Слика 5.12. Почетен екран на апликацијата

Во левиот дел од менито се наоѓа линкот Дома кој води кон почетната страница на апликацијата, каде е отворено полето за пребарување. Доколку корисникот е од тип Администратор се појавува и истоимената група на функции. Менито Подесувања се појавува за корисници од дадена компанија кои може да ги конфигурираат и менуваат поставувањата за начинот на индексирање и пребарување во нивната компанија. Останатите функции во системот се кориснички и се видливи за сите останати корисници во системот.

5.4.3 Администраторски функции

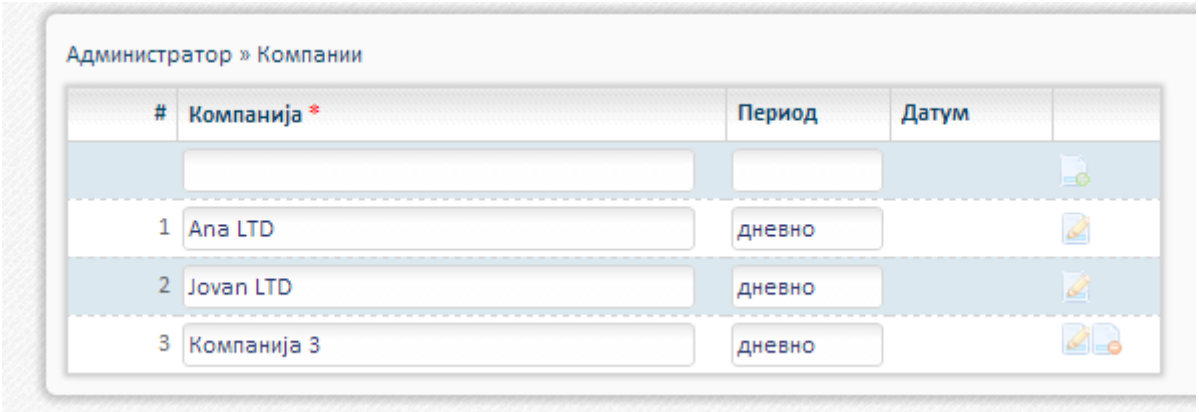
Администраторски функции се функциите Компании и Корисници (Слика 5.13).



Слика 5.13. Администраторски функции

Компании

Функцијата Компании се користи за менаџирање на податоци за компаниите т.е. клиентите кои ги користат функционалностите на Артисофт Пребарувачката Машина. На Слика 5.14. е прикажан екранот преку кој се внесуваат и ажурираат називот на компанијата и периодот на освежување на главниот индекс. Копчето за бришење на компанијата се појавува се додека истата нема активни корисници.



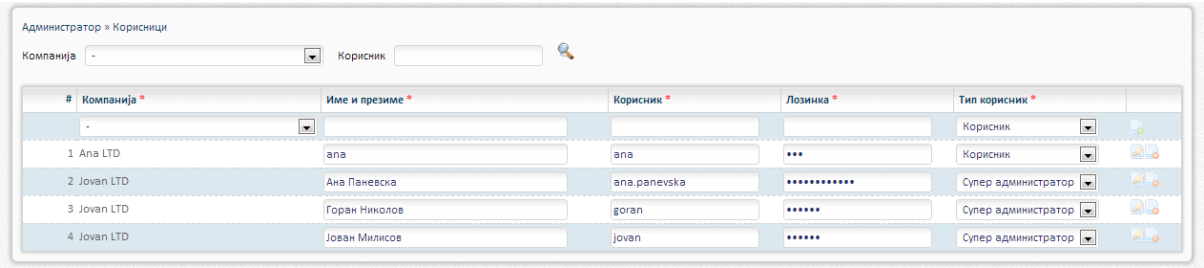
Слика 5.14. Функција Компании

Корисници

Функцијата Корисници (Слика 5.15) се користи за менаџирање на корисниците по различни компании. Во горниот дел од функцијата има филтри за пребарување:

Компанија и Корисник. Во долниот дел од функцијата се прикажува форма во која се излистани сите корисници според пополнетите филтри заедно со Реден број, Компанија, Име и презиме, Корисничко име, Лозинка и Тип на корисник.

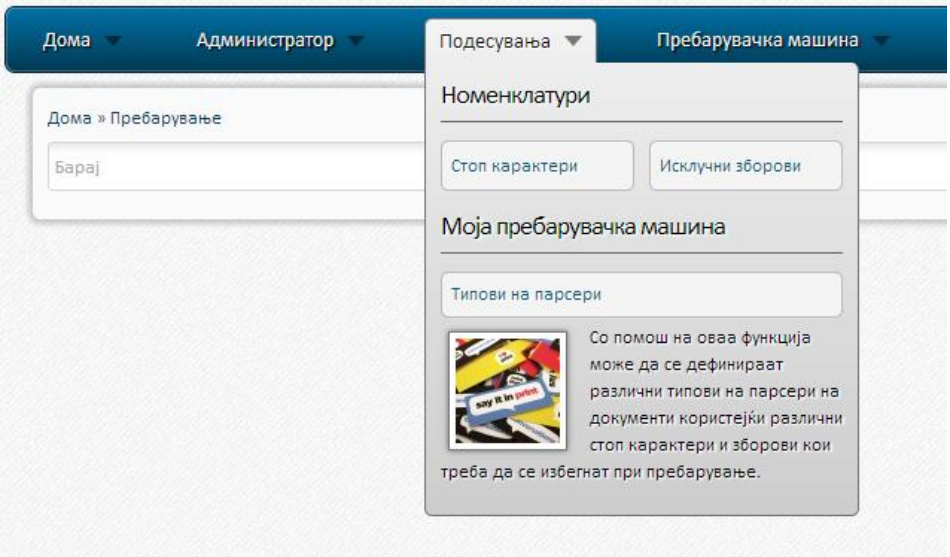
Преку истата форма се ажурираат податоци за постоечките корисници, се бришат постоечките и се додаваат нови.



Слика 5.15. Функција Корисници

5.4.4 Конфигурациски функции

Конфигурациски функции се функциите Стоп карактери, Исклучни зборови, Типови на парсери (Слика 5.16), како и фунцијата Документи која е дел од менито Пребарувачка машина.

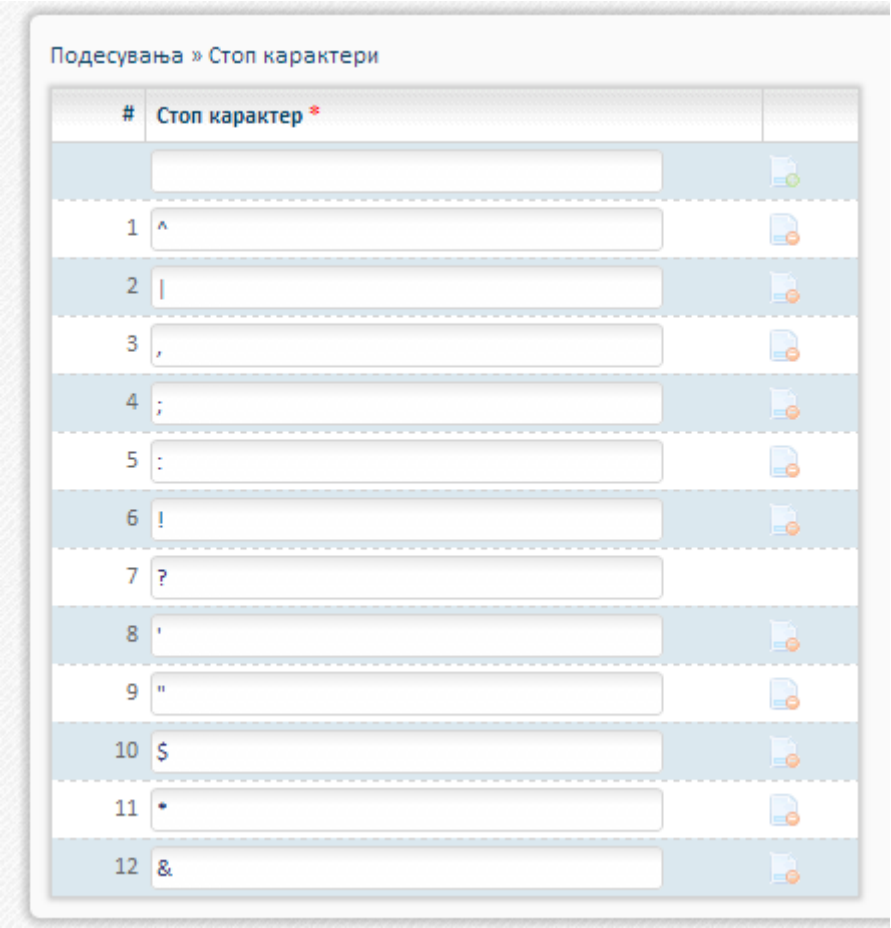


Слика 5.16. Администраторски функции

Стоп карактери

Функцијата Стоп карактери (Слика 5.17) се користи за менаџирање на стоп карактерите кои се користат по различни компании. За истите да бидат земени во предвид при парсирање на текстуалната содржина на документи и упити потребно е да се креира парсер кој ќе ги вклучи потребните стоп карактери. Оваа функција е само

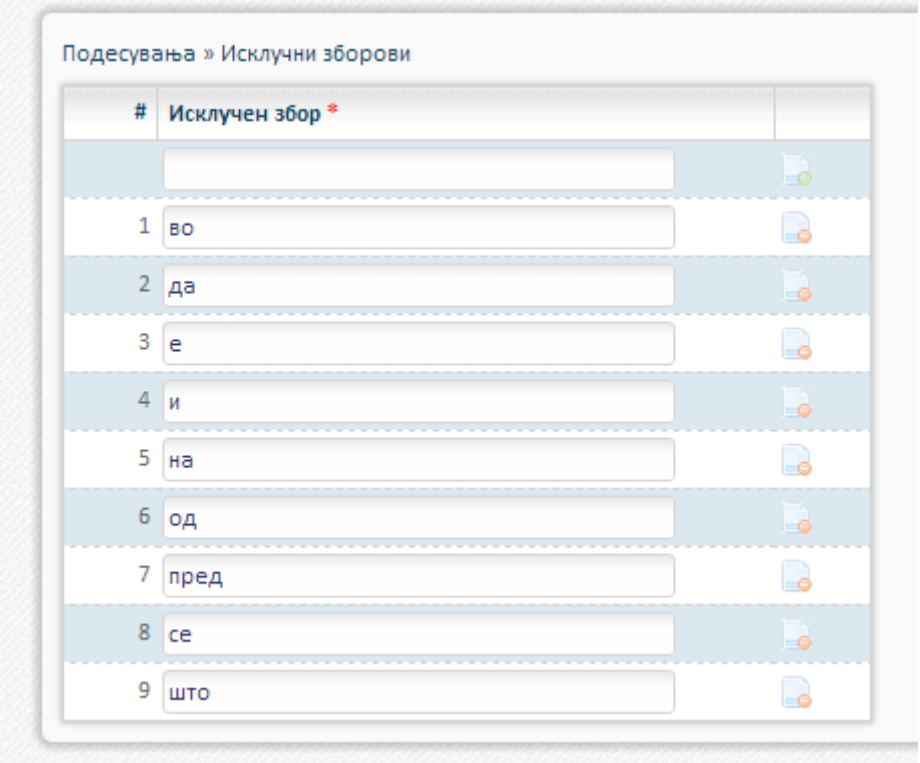
номенклатура за внес, додека парсерот се креира преку друга функција која ќе ја опишеме во продолжение. Оние стоп карактери кои се користат кај одреден парсер не може да се избришат.



Слика 5.17. Функција Стоп карактери

Исклучни (стоп) зборови

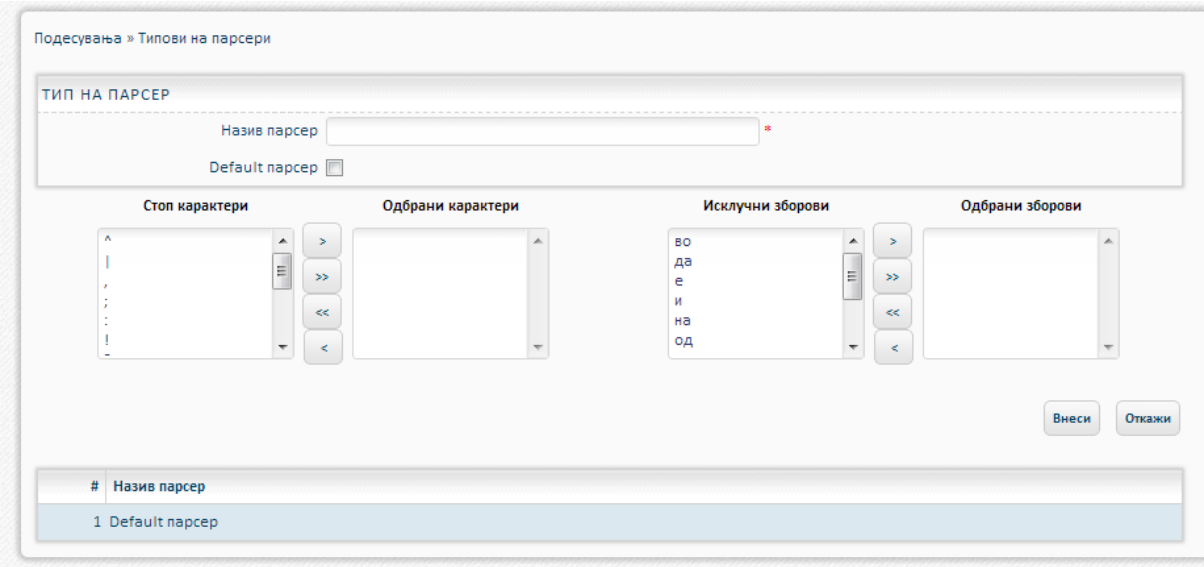
Функцијата Стоп зборови (Слика 5.18) се користи за менаџирање на стоп зборовите кои се користат по различни компании. За истите да бидат земени во предвид при парсирање на текстуалната содржина на документи и упити потребно е да се креира парсер кој ќе ги вклучи потребните стоп зборови. Оваа функција е само номенклатура за внес, додека парсерот се креира преку друга функција која ќе ја опишеме во продолжение. Оние стоп зборови кои се користат кај одреден парсер не може да се избришат.



Слика 5.18. Функција Исклучни зборови

Типови на парсери

Функцијата Типови на парсери се користи за внес и ажурирање на податоци за различни типови на парсери. На Слика 5.19 е дадена формата за ажурирање на веќе внесен тип на парсер.



Слика 5.19. Функција Тип на парсер

Во долниот дел од екранот се излистани сите постоечки типови на парсери од каде со клик на линкот се пополнува формата во горниот дел од екранот за моментално

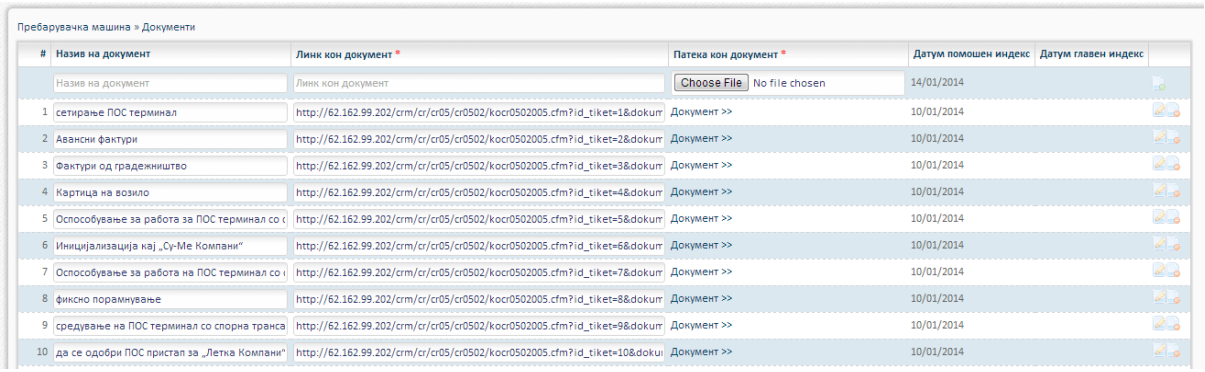
селектираниот парсер. За секој тип на парсер се пополнува назив и индикатор дали истиот е default парсер. Default парсер е оној кој се употребува при индексирање и пребарување на документи доколку не е специфицирано кој тип на парсер да се користи. Секако, за секој тип на парсер се избираат и записи од номенклатурите стоп карактери и исклучни (стоп) зборови кои дефинираат на кој начин конкретниот тип на парсер ќе го процесира текстот.

Документи

Функцијата Документи се користи за евиденција на внесените документи. За секој документ се води назив, URL адреса и датуми на индексирање во помошниот и главниот индекс. Излистаните документи (Слика 5.20) се всушност веќе индексираните документи.

Освен преку екстерни сервиси за полнење на индексот, документи може да се додаваат и преку оваа функција при што за секој документ се внесува називот, URL адреса и се upload-ува самиот документ кој треба да се индексира. Потоа во позадина се повикува сервисот кој ќе ја индексира содржината на документот.

Функцијата се користи и за бришење на документи. Во моментот кога ќе се кликне на копчето Избриши само се пополнува индикатор дека документот треба да се избрише од индексот, а потоа на одреден период, односно пред освежување на главниот индекс, документите се бришат од истиот, како и записот во базата на податоци.



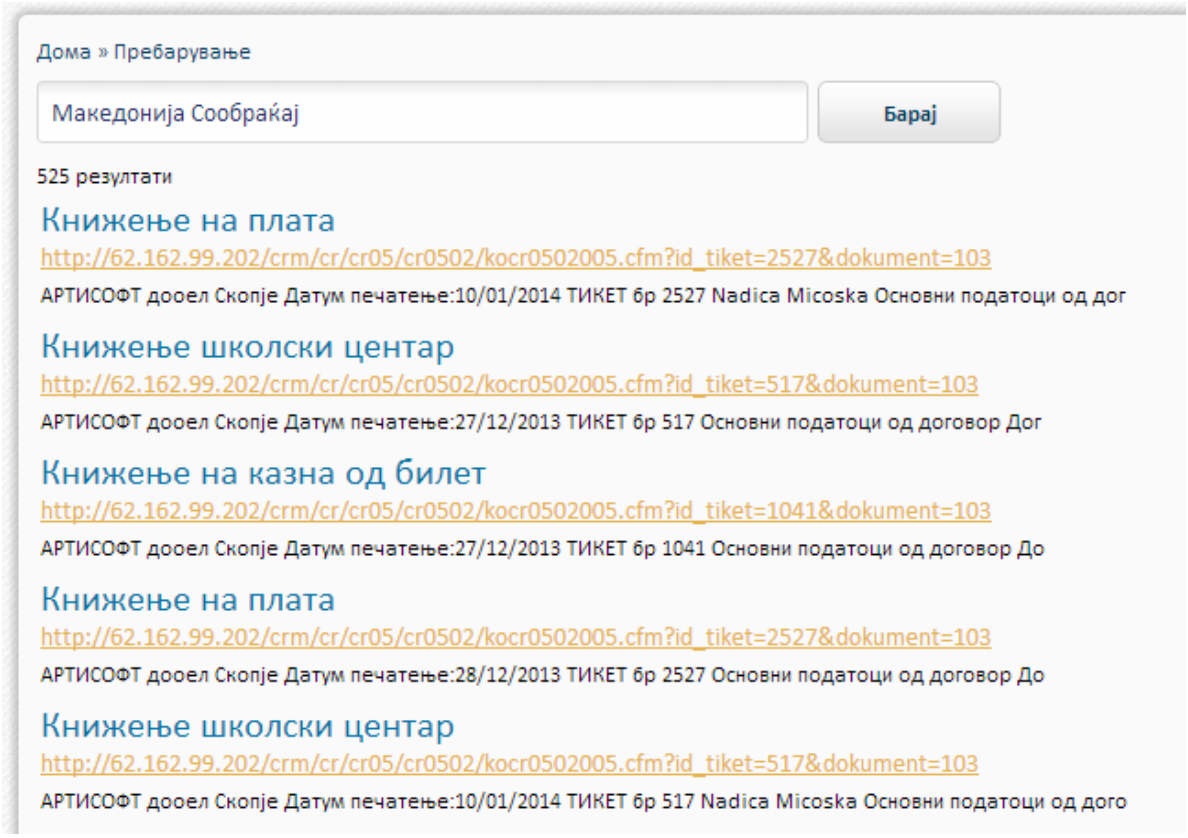
Слика 5.20. Функција Документи

5.4.5 Кориснички функции

Кориснички функции се: Функцијата Пребарување (Дома), Најчести пребарувања и Последни пребарувања.

Пребарување

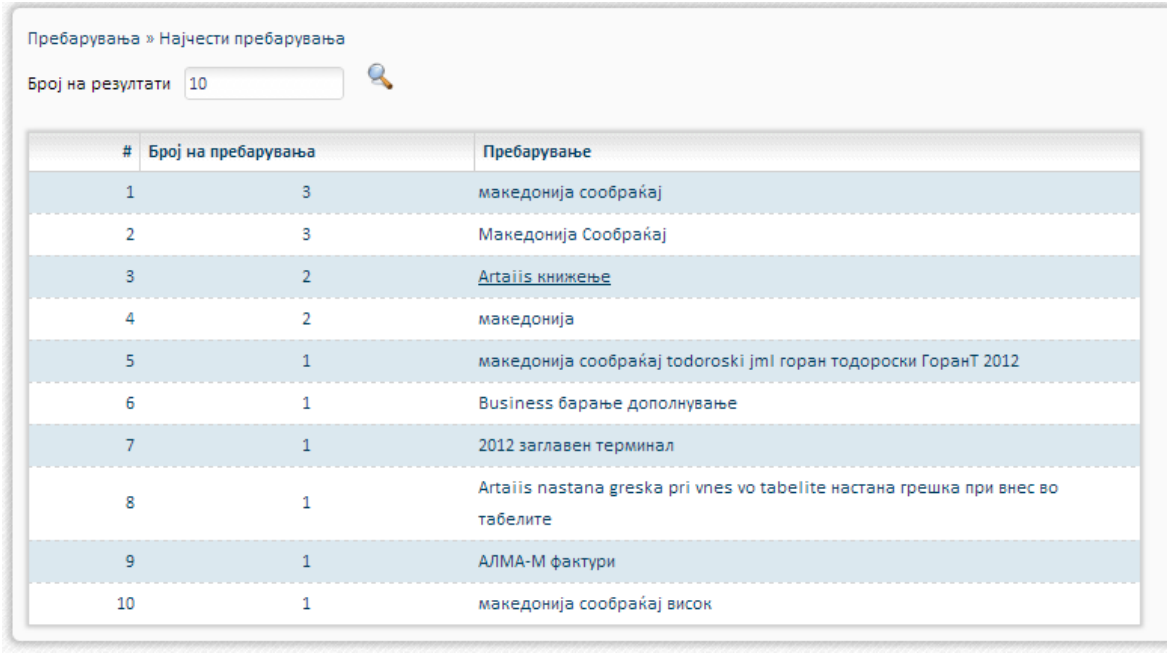
Функцијата Пребарување е почетната функција т.е. екран кој се појавува по најавување во апликацијата. На Слика 4.21 е даден екранот на кој се прикажани резултатите од пребарување на клучните зборови “Македонија Сообраќај”. Функцијата е релативно едноставна. Формата содржи едно поле во кое се внесуваат клучните зборови. Откако ќе се пополнат со клик на “Enter” или копчето Барај се прикажува бројот на резултати, како и сите резултати од пребарувањето со Наслов на документ, URL адреса и почетните зборови од документот. Идејата е оваа функционалност да може да се вметне во било која апликација. Упатство за начинот на вметнување е даден во менито Помош и ќе биде подетално објаснето во конкретна имплементација во сегментот 5.5.



Слика 5.21. Функција Пребарување

Најчести пребарувања

Функцијата Најчести пребарувања е дел од менито Пребарувања. Функцијата се користи за историјат на најчестите пребарувања. Иницијално се прикажуваат 10-те најчести пребарувања за најавениот корисник, како и бројот на пребарувања за конкретната комбинација на клучни зборови (Слика 5.22), иако овој број може да се промени. Со клик на линкот од дадено пребарување функцијата редиректира кон функцијата Пребарување со веќе пополнети вредности за клучните зборови и веќе прикажани резултати.

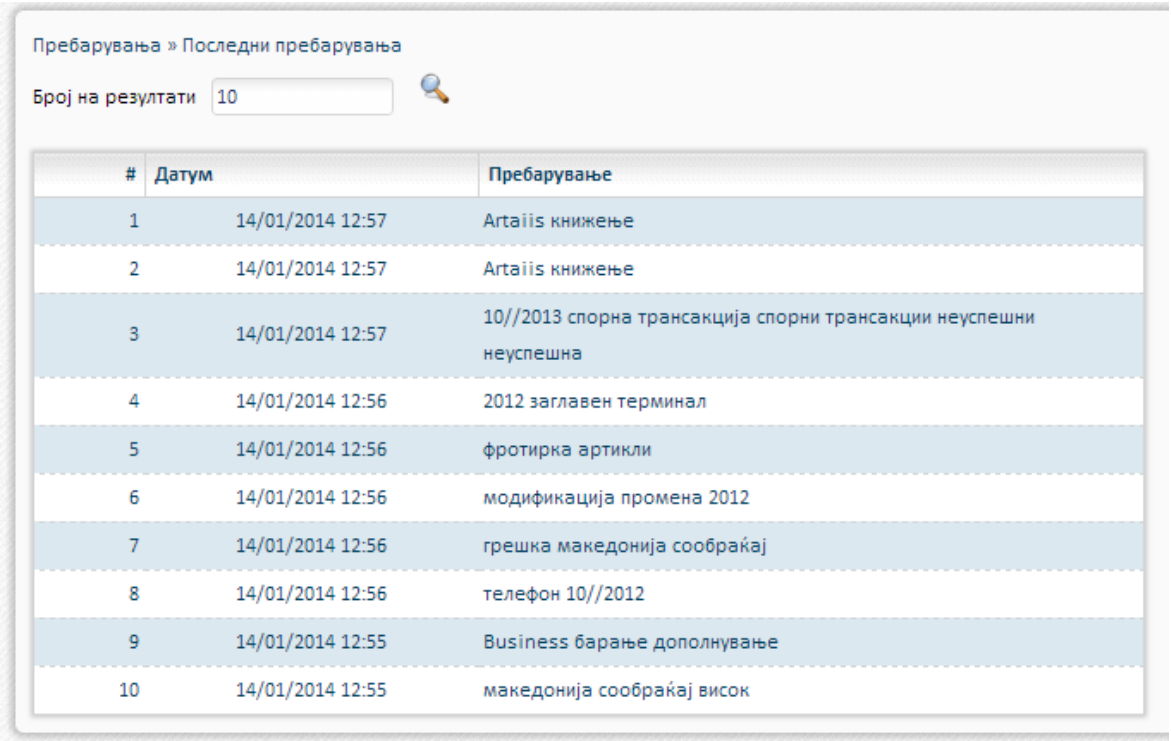

Пребарувања » Најчести пребарувања

Број на резултати 10

#	Број на пребарувања	Пребарување
1	3	македонија сообраќај
2	3	Македонија Сообраќај
3	2	Artaiis книжење
4	2	македонија
5	1	македонија сообраќај todoroski jml горан тодороски ГоранТ 2012
6	1	Business барање дополнување
7	1	2012 заглавен терминал
8	1	Artaiis nastana greska pri vnes vo tabelite настана грешка при внес во табелите
9	1	АЛМА-М фактури
10	1	македонија сообраќај висок

Слика 5.22. Функција Најчести пребарувања

Последни пребарувања


Пребарувања » Последни пребарувања

Број на резултати 10

#	Датум	Пребарување
1	14/01/2014 12:57	Artaiis книжење
2	14/01/2014 12:57	Artaiis книжење
3	14/01/2014 12:57	10//2013 спорна трансакција спорни трансакции неуспешни неуспешна
4	14/01/2014 12:56	2012 заглавен терминал
5	14/01/2014 12:56	фротирка артикли
6	14/01/2014 12:56	модификација промена 2012
7	14/01/2014 12:56	грешка македонија сообраќај
8	14/01/2014 12:56	телефон 10//2012
9	14/01/2014 12:55	Business барање дополнување
10	14/01/2014 12:55	македонија сообраќај висок

Слика 5.23. Функција Последни пребарувања

Функцијата Последни пребарувања е дел од менито Пребарувања. Функцијата се користи за историјат на последните пребарувања. Иницијално се прикажуваат 10-те последни пребарувања за најавениот корисник, како и датумот и времето на пребарување (Слика 5.23), иако овој број може да се промени. Со клик на линкот од

дадено пребарување функцијата редиректира кон функцијата Пребарување со веќе пополнети вредности за клучните зборови и веќе прикажани резултати.

5.5 Имплементација на Артисофт Пребарувачка Машина во ЕРП системот АртАИИС

ЕРП системите на компаниите се централно место каде се чуваат сите документи потребни за функционирањето на истите. Недостигот на ефикасна и ефективна пребарувачка машина која ќе пребарува низ множество на документи од различни категории кај ваквите системи е клучен проблем. Пребарувањето во ЕРП системите најчесто се врши со филтрирање преку полиња (атрибути) каде корисникот е потребно да знае точна информација за идентификување на документот или пребарувањето резултира во голема листа на документи, подредени по нерелевантни фактори за оној кој пребарува. Од овие причини сметаме дека дизајнирање на пребарувачка машина за т.н. физички непостоечки документи (фактури, требовници, испратници и сл.) во рамки на еден ЕРП систем чии податоци се запишуваат во повеќе табели во база на податоци значително ќе ја зголеми ефикасноста на барањето на податоци и ќе го намали времето кое се троши за добивање информации.

5.5.1 АртАИИС ЕРП

АртАИИС е ЕРП системот на Артисофт кој генерира 50-тина различни типови на документи. Пребарувањето се врши низ посебни функции за секој различен тип на документи со предефинирани филтри селектирани според проектантските анализи и одлуки при иницијалното проектирање на системот или пак при редизајн, секако со ограничувачки фактори. Колку и да е добро испроектиран даден систем, не би можеле да се земат во предвид сите можни сценарија, промени и потреби на корисниците, па во даден момент системот не е доволно флексибилен и не ги задоволува сите барања на корисниците.

Понапредна верзија на статичко пребарување низ различните видови на документи е т.н. Генератор на извештаи – функционалност која им овозможува на корисниците самостојно да дефинираат низ кои податоци (табели во база на податоци) ќе пребаруваат, потоа филтрите по кои ќе се пребарува, како и информациите кои ќе се прикажат на екран при листање на резултатите од пребарувањето. Секое индивидуално конфигурирано пребарување може да се зачува и да се користи во било кое време

менувајќи ги вредностите на филтрите со што се добиваат различни резултати од пребарувањето. Ваквиот Генератор на извештаи всушност се користи при споредба и евалуација на АртАИИС наспроти Пребарувачката Машина.

Иако со помош на генераторот на извештаи се постигнува поголем степен на флексибилност, сепак останува како нерешено прашање фактот дека корисникот треба да има познавање на структурата на податоците во документите и поврзаноста на табелите во базата на податоци за да може да пребарува. Од друга страна пак, корисникот повторно мора да знае точно по кои филтри треба да пребарува за да ги добие очекуваните резултати. Ваквото предзнаење е практично невозможно и вообичаено овие извештаи ги изработуваат програмерите на системот на претходни специфицирани барања на корисниците.

Додека вебот претставува колекција на документи, кадешто многу често, особено при пребарување на новости, висок критериум за пребарување е документите да бидат што е можно понови, ова не е случајот со пребарување низ колекции документи од типот на оние кои се чуваат во ЕРП системите. Самите корисници кои ги генерираат документите во овие системи, толку добро ги познаваат истите, што може доволно брзо да дојдат до податоци и документи кои се релативно нови.

Она што е проблем во ЕРП системите е пронаоѓање на документи кои се постари знаејќи релативно малку информации за структурата на податоците, типот на документи низ кои се пребарува, како и атрибутот според кој треба да се пребарува. Многу често пребарувањето е според претходно запамтена одредена информација, но не се знае каде е таа информација, во кој тип на документ, во кој период, а уште помалку во кој од атрибутите за даден документ.

Од досега набројаните причини евидентна е потребата од имплементација на ефикасен текст базиран пребарувачки алгоритам во ЕРП системите кој едноставно и брзо ќе пребарува низ сите видови на документи (и виртуелни содржани низ табелите на системот и вистински содржани низ базите на документи), односно народски кажано – ќе пребарува низ “баби и жаби”.

5.5.2 Избор на документи за имплементација во Пребарувачка Машина

Веќе утврдивме зошто ЕРП системите се интересни за имплементација на пребарувачка машина. Долгорочно, целта е да се воведе можност за пребарување низ сите типови на документи во дадена компанија без разлика од кој систем потекнуваат,

од која категорија се и кој ги креирал. Секако, овој процес мора да биде итеративен. Најпрво, мора да се изберат неколку категории на документи кои ќе бидат процесирани и индексирани од страна на пребарувачката машина и врз кои ќе може да се пребарува. Потоа, се избира период во кој се генерирани истите. Овој чекор не е задолжителен доколку нема голем број на документи во дадена категорија. Сепак, за компании кои генерираат илјадници документи на годишно ниво, овој чекор мора да биде земен во предвид.

Имплементацијата на Пребарувачката Машина врз АртАИИС е само една од фазите на имплементација во Артисофт и е сеуште во прогрес. Целата имплементација се планира да се одвива постепено во неколку фази:

- *Фаза 1* - Оваа фаза вклучува имплементација и тестирање на пребарувачката машина за сите документи од тип *Тикети* во Артисофт т.е. во АртАИИС. Во продолжение ќе бидат резимирани повеќе детали од оваа фаза која е веќе завршена.
- *Фаза 2* - Иако одредени модификации се имплементирани и во Фаза 1, оваа фаза се концентрира на оптимизација на имплементацијата, како и предлози за евентуални модификации и подобрувања на самиот алгоритам за пребарување. Ваквите чекори се подетално опишани во Заклучокот на магистерскиот труд.
- *Фаза 3* - Имплементација на пребарувачка машина врз документи од тип: Фактури, Договори, Понуди.
- *Фаза 4* – Имплементација на пребарувачка машина врз документи од тип Задачи од АртНЕТ системот на Артисофт за колаборација и проектен менаџмент и ДМС системот (Систем за менаџирање на документи).

Првата фаза од горенабројаните е веќе имплементирана и се однесува на воведување на можност за пребарување на сите документи од АртАИИС од тип Тикети. *Тикет* претставува документ кој се генерира за секоја активност во рамки на компанијата која подразбира интеракција со клиент на истата. Секоја пријава на проблем, барање за модификација, оперативна поддршка и сл., резултира со документ Тикет во ЦРМ (CRM – Customer Relationship Management) модулот од АртАИИС. Тикетите се интересни документи за пребарување бидејќи содржат повеќе описни податоци внесени од страна на клиентите, како опис и манифестирање на проблем, и од страна на вработените кои го разрешуваат проблемот, како што се причина за појава на проблем, мислење, опис на решение на проблемот и сл.

Целта со завршувањето на оваа фаза е имплементација на пребарувачката машина за овој тип на документи, тестирање и евалуација на системот и изведување на заклучоци од истата во форма на упатства за понатамошна имплементација, како и предлог модификации за подобрување и оптимизација на целиот систем. Врз основа на искуствата и заклучоците понатаму би се извршувале и активностите предвидени во останатите фази.

5.5.3 Чекори при имплементација на системот

Во овој сегмент ќе бидат резимирани сите чекори потребни за имплементација на можност за индексирање и пребарување на документи од АртАИИС од тип Тикети преку Артисофт Пребарувачката Машина (Фаза 1). Овие чекори се генерализирани и важат при имплементација на Пребарувачката машина над било која колекција на документи од било каква категорија.

АртАИИС како ЕРП систем веќе поседува можност за пребарување на Тикети преку претходно опишаната функција Генератор на извештаи. Сепак, како што утврдивме и претходно, предноста од имплементација на Пребарувачка машина е евидентна. И покрај тоа, процесот на имплементација не смее да биде комплексен, во спротивно, корисникот (во овој случај Артисофт) не би се нафатил на имплементација на Пребарувачката машина. Затоа, од клучно значење е чекорите за имплементација да бидат брзи и едноставни.

1) *Чекор 1: Поставување на компанија* – Првиот чекор е евиденција на компанијата која ќе ја користи Пребарувачката машина. Лице во Артисофт кое е задолжено за менаџирање на компаниите во системот ги пополнува податоците за компанијата (во овој случај повторно Артисофт) и креира иницијален корисник за истата кој ќе може да се најави во системот и да добие упатства како понатаму треба да постапи за да се заврши процесот на имплементација. За таа цел се користат претходно опишаните функции за администрација – Компании и Корисници.
2) *Чекор 2: Внес и мапирање на корисници* – Секое пребарување се евидентира во историјат на пребарувања за корисникот кој пребарувал. Според досегашните чекори, компанијата во моментот има само еден иницијален корисник од тип Администратор кој може да ги конфигурира параметрите на системот кои би влијаеле на индексирањето и пребарувањето на документи. Истиот корисник има и

можност за користење и на корисничките функции, од кои една е и основната – Пребарување. Според тоа, креирање на дополнителни корисници е незадолжителен чекор, доволно е компанијата да има еден корисник. Сепак, доколку компанијата има потреба да води евиденција кој корисник што пребарувал, потребно е креирање на повеќе корисници. Оваа активност може да ја изврши лице вработено во Артисофт или пак самиот корисник директно од функцијата Корисници. Од друга страна, постои и јавно достапен веб сервис, т.е. метод за внес на корисници. Овој метод се повикува од екстерниот систем, АртАИИС ЕРП, при што сервисот враќа идентификатор на креираниот корисник. На овој начин овозможуваме мапирање на корисниците во АртАИИС ЕРП и корисниците за компанијата во Пребарувачката машина. Ваквите корисници може да се најават на апликацијата Пребарувачка машина или пак да пребаруваат директно од веб интерфејсот на АртАИИС ЕРП системот, за што е потребен дополнителен чекор за интеграција на корисничкиот интерфејс, кој ќе биде објаснет во продолжение.

3) *Чекор 3: Конфигурација на системот* – Задолжителен чекор пред да се индексираат документите е конфигурација и параметризација на начинот на процесирање на содржината на документите. Овој чекор опфаќа пополнување на номенклатурите Стоп карактери и Исклучни зборови, како и креирање на потребните типови на парсери. Иницијално се креира еден тип на парсер кој треба да се означи како default парсер, кој ги вклучува дефинираните Стоп карактери и Исклучни зборови. Понатаму, доколку има потреба, возможно е дефинирање на повеќе типови на парсери врз основа на категориите на документите кои се процесираат. Оваа потреба се согледува дури при стекнато искуство во користење на целиот систем. Затоа, овој чекор иницијално подразбира само креирање на еден тип на парсер. Типот на парсер кој го креиравме за целите на имплементација на Пребарувачката машина во АртАИИС е означен за default и ги вклучува стоп карактерите ^|,;:!?'"$*&/\., како и исклучните зборови *во, да, до, е, и, иако, итн, каде, како, на, но, од, по, покрај, пред, се, сепак, сл, т.е., што*, итн.

4) *Чекор 4: Полнење на индекс* – Веќе ги објаснивме компонентите *documentCrawler* и *documentCreator* кои се користат за обработка на документите кои треба да се индексираат и повикување на компонентата Index Builder која ќе ги индексира. Компонентата *documentCreator* е пригодна доколку документите физички постојат на дадена мрежно достапна локација. Во нашиот случај документите (Тикетите) се виртуелни, т.е. се комбинација од податоци кои се чуваат во табели во релациона

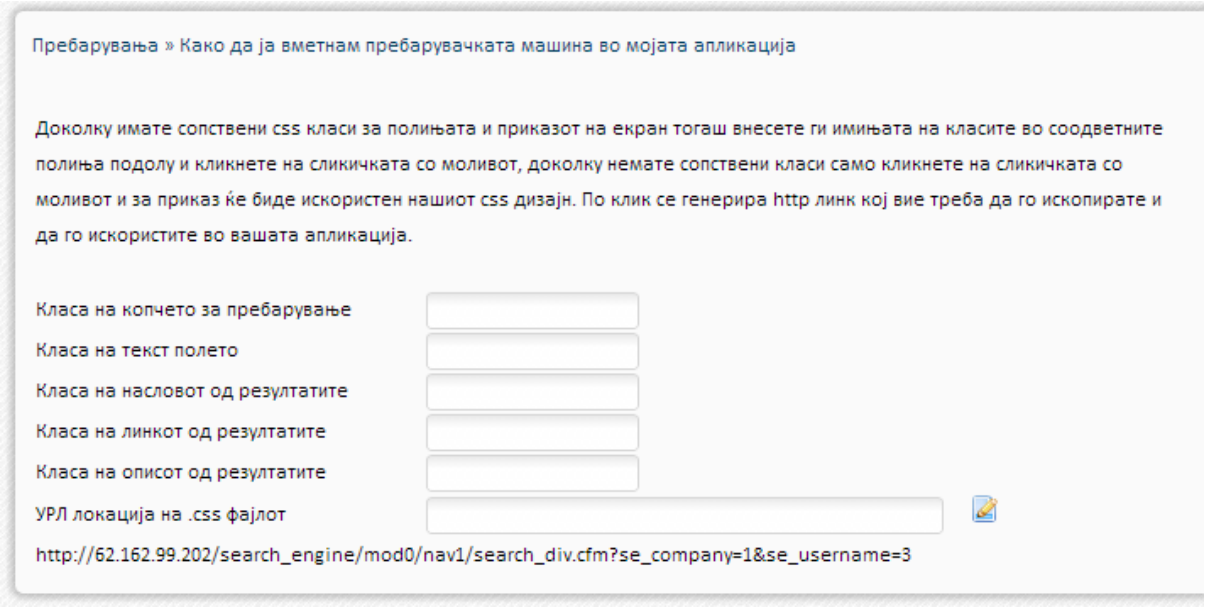
база на податоци. Во овој случај мора да креираме *docProxy* компонента, односнокод кој ќе ги креира документите кои сакаме да ги индексираме. Секој систем во кој се чуваат виртуелни документи има можност за нивно печатење при што документот се генерира во некаков формат (најчесто pdf).

Наједноставен начин за креирање на оваа *docProxy* компонента е процесирање на сите документи кои треба да се индексираат со пишување на едноставни sql скрипти или пак повикување на постоечки и повикување на модулот за печатење на истите при што документот физички се зачувува. По зачувување *docProxy* ја повикува *documentCreator* компонентата (сервисот) која ќе го индексира документот. На крај *docProxy* го брише генерираниот документ од системот на датотеки. Овој принцип го искористивме при полнење на индексот со Тикетите од АртАИИС ЕРП системот, при што индексиравме 2775 тикети кои постоеа во моментот.

Освен полнење на индексот со постоечките, треба да се земе во предвид и индексирање на новонастанати документи. Оваа активност не се разликува од претходната. Имено, она што треба да се направи е да се дополни *docProxy* компонентата така да го креира физички само оној документ кој се внесува во моментот и да ја повика *documentCreator* компонентата.

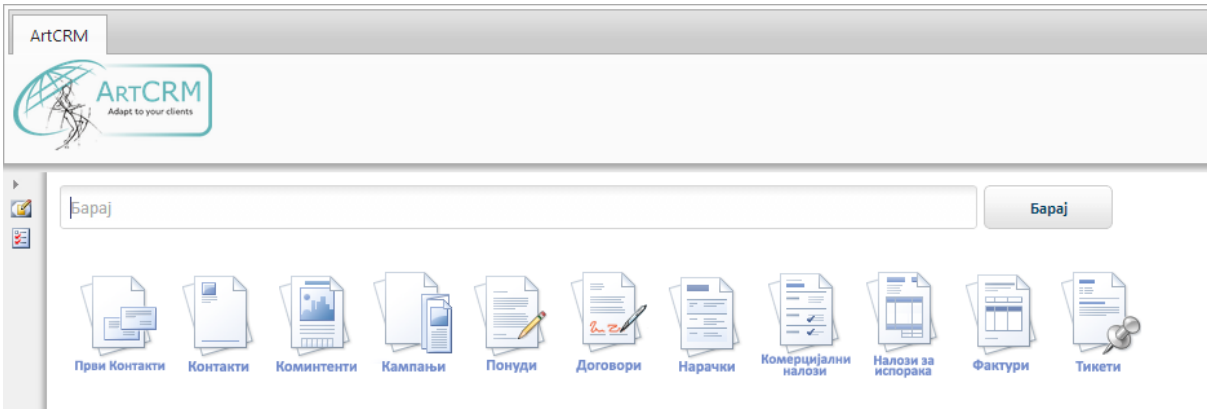
Овој чекор е всушност најкомплексен, но треба да биде релативно едноставен за оној кој ја познава структурата на документите кои сака да ги индексира. Упатството за користење и имплементација на Пребарувачката Машина детално ги опишува методите и сервисите кои треба програмски да се повикаат во овој чекор.

5) *Чекор 5: Интеграција на кориснички интерфејс* – Овој чекор подразбира воведување на можност за пребарување низ индексираните документи преку апликацијата на самиот корисник, во овој случај АртАИИС ЕРП системот. За таа цел во истата се уфрлува едноставен код кој може да се конфигурира преку самата Пребарувачка машина користејќи ја функционалноста *Како да ја вметнам пребарувачката машина во мојата апликација* која се наоѓа во Помош менито на Пребарувачката машина (Слика 5.24). Преку овој код се идентификува компанијата и корисникот кој пребарува. Имено, параметарот *se_username=3* во овој случај се однесува на моментално најавениот корисник. Ова би функционирало за било кој корисник во АртАИИС ЕРП, но доколку сакаме да се води евиденција кој корисник што пребарувал, потребно е корисниците да се мапираат на одреден начин во АртАИИС и вредноста на идентификаторот (која во овој случај е 3) да се чува за секој корисник во АртАИИС кој има кориснички налог во Пребарувачката Машина.



Слика 5.24. Конфигурација на код за вградување на кориснички интерфејс

Дополнително, освен конфигурација на корисникот, возможно е и имплементирање на сопствен css дизајн на елементите кои се прикажуваат во интерфејсот. За таа цел потребно е во формата да се внесат податоци како URL адреса на .css датотеката, како и називите на класите за текст полето за пребарување, копчето, насловот, линкот и описот на резултатите. Во спротивно дизајнот на интерфејсот ќе биде превземен од оној во рамки на апликацијата Пребарувачка машина, како што е случајот на Слика 5.25., каде интерфејсот е вграден во АртАИИС ЕРП системот – ArtCRM модулот.



Слика 5.25. Интеграција на кориснички интерфејс во АртАИИС ЕРП (ArtCRM)

6 Евалуација и тестирање на моделот

Во оваа секција ќе ги резимираме резултатите од евалуацијата и тестирањето на практичната имплементација на Артисофт Пребарувачката Машина врз ЕРП системот АртАИИС. Ќе биде резимирана методологијата која се користи за евалуација на системот, најпрво со детален опис на тест колекцијата на документи врз кои ќе се спроведат мерења, потоа ќе ги опишеме метриките и мерките за ефективност и ефикасност на системот, за на крај да ги прикажеме резултатите кои се добиени за овие мерки врз формираната колекција. Оваа секција исто така содржи и дискусија на добиените резултати, како и заклучок од истите.

6.1.1 Опис на тест колекција

Во оваа секција ќе ги опишеме сите делови од формираната тест колекција. Веќе споменавме од кои сегменти се состои една тест колекција во поглавје **Error! eference source not found.**:

- *Колекција на документи* е множество од документи кои ќе се индексираат и врз кои ќе се извршува пребарување и мерење;
- Колекција, т.е. тест множество од информациски потреби (пребарувања, упити);
- Множество на релевантни документи за секој упит во тест множеството на информациски потреби.

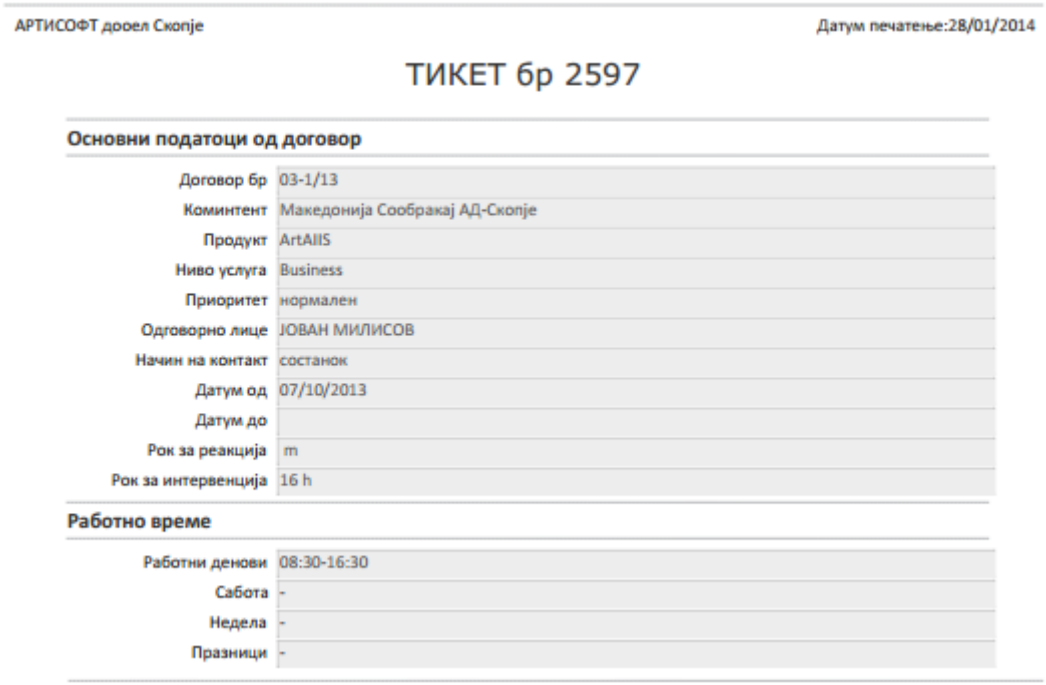
На Слика 5.26. се прикажани сегментите на тест колекцијата.

Слика 5.26. Сегменти на тест колекција

Тест колекцијата може да биде дел од постоечки стандардни тест колекции кои често се користат за мерење на ефикасност и ефективност на IR системи (**Error! Reference ource not found.**). Во нашиот случај идејата е Артисофт Пребарувачката Машина да се имплементира врз сопствени вистински документи, како и вообичаени случаи на информациски потреби изразени преку упити, за на крај да добиеме колекција на релевантни резултати. Во продолжение ќе биде опишан секој од нив подетално.

6.1.1.1 Опис на колекцијата на документи

Како што веќе споменавме претходно колекцијата на документи е множество од сите документи од АртАИИС од тип Тикети. Тикет претставува документ кој се генерира за секоја активност во рамки на компанијата која подразбира интеракција со клиент на истата. Секоја пријава на проблем, барање за модификација, оперативна поддршка и сл., резултира со документ од овој тип во ЦРМ (CRM – Customer Relationship Management) модулот од АртАИИС. Тикетите содржат повеќе описни податоци внесени од страна на клиентите, како опис и манифестирање на проблем, како и информации кои се внесуваат од страна на вработените кои го разрешуваат проблемот, како што се причина за појава на проблем, мислење, опис на решение на проблемот и сл. На Слика 5.27 и Слика 5.28 се прикажани извадоци од документ од тип Тикет.



АРТИСОФТ дооел Скопје Датум печатење:28/01/2014

ТИКЕТ бр 2597

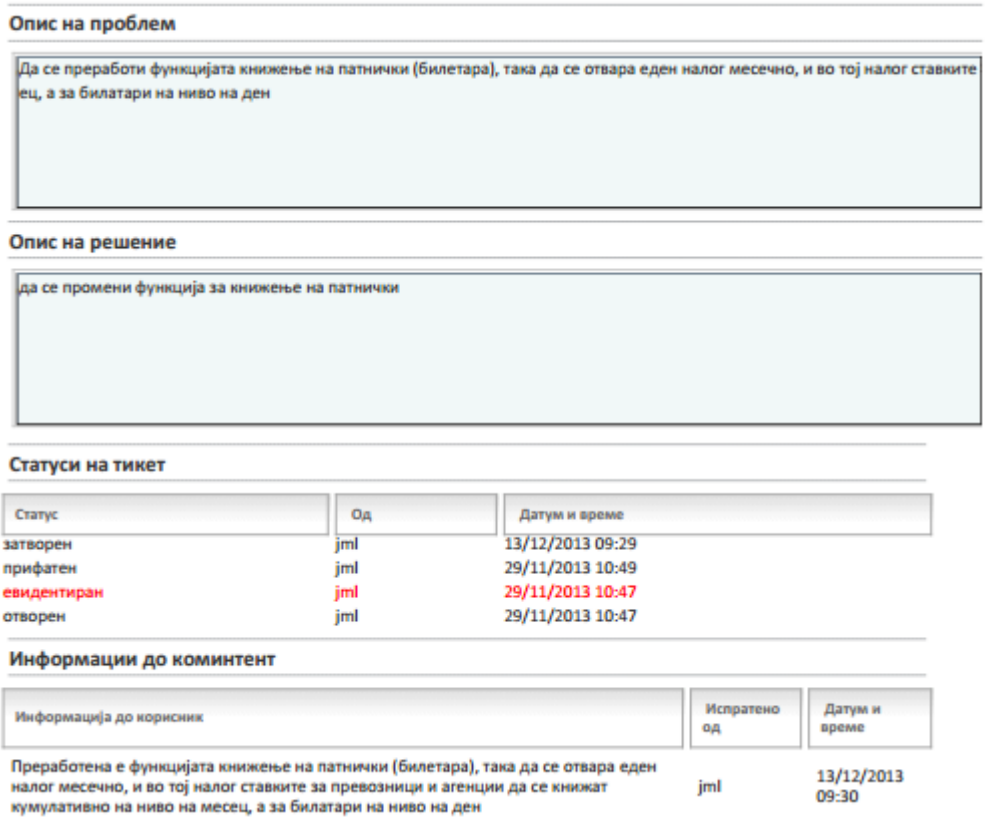
Основни податоци од договор

Договор бр	03-1/13
Коминтент	Македонија Сообраќај АД-Скопје
Продукт	ArtAIIS
Ниво услуга	Business
Приоритет	нормален
Одговорно лице	ЈОВАН МИЛИСОВ
Начин на контакт	состанок
Датум од	07/10/2013
Датум до	
Рок за реакција	m
Рок за интервенција	16 h

Работно време

Работни денови	08:30-16:30
Сабота	-
Недела	-
Празници	-

Слика 5.27. Извадок од документ (1)



Слика 5.28. Извадок од документ (2)

Процесот на градење на колекцијата на документи се состои од два чекора, на начин опишан претходно во **Error! Reference source not found.**:

1. *Креирање на компонента/сервис docProxy* – Овој сервис е генериран од страна на лице кое има познавање во областа на ЦРМ. Познавањето на начинот на кој функционира Артисофт Пребарувачката Машина е незадолжително. Сервисот ги изминува сите документи од тип Тикет во базата на податоци на Артисофт и за истите ја генерира содржината во форма на стринг.
2. *Повикување на компонента/сервис docCreator* - За секој документ се повикува овој сервис кој истиот ќе го индексира. Овој сервис е еден од екстерните сервиси на Артисофт Пребарувачката Машина за комуникација со надворешни системи. Екстерните сервиси се детално опишани заедно со придружните методи и влезните и излезните аргументи, така што може да се повикуваат од било кој систем, без разлика која технологија се користи за развој на системот, се додека има поддршка за повикување на веб сервиси. docCreator се повикува за сите претходно генерирани документи. Исто така, може да се вметне во

програмската логика за креирање на Тикети, така што со генерирање на секој нов тикет истиот веднаш ќе се вметне во индексот. Од друга страна, овој процес може да биде и периодичен, со што документите во двата система нема да бидат исти на број во секој момент. За целите на пребарување, евалуација и мерење на ефикасноста на двата система, сите тикети во базата на податоци на Артисофт мора да бидат индексирани.

Во продолжение, во Табела 5.1. се прикажани основни статистики за колекцијата на документи (Тикети):

Табела 5.1. Статиситики за колекцијата на документи

Бр. на документи	2908	**Бр. на зборови**	363166
Просечен бр. на зборови во документ	124.8	**Големи на индекс на хард диск**	1.12MB
Време на генерирање на документ	60ms	**Време на индексирање на документ**	422ms

Времето на индексирање зависи и од форматот во кој е документот. Во нашиот случај содржината е веќе во стринг формат. За различни формати на документи (word, pdf, excel, txt итн.) ќе биде потребно дополнително време за да се вчита и испроцесира документот, односно неговата содржина да се добие во стринг формат.

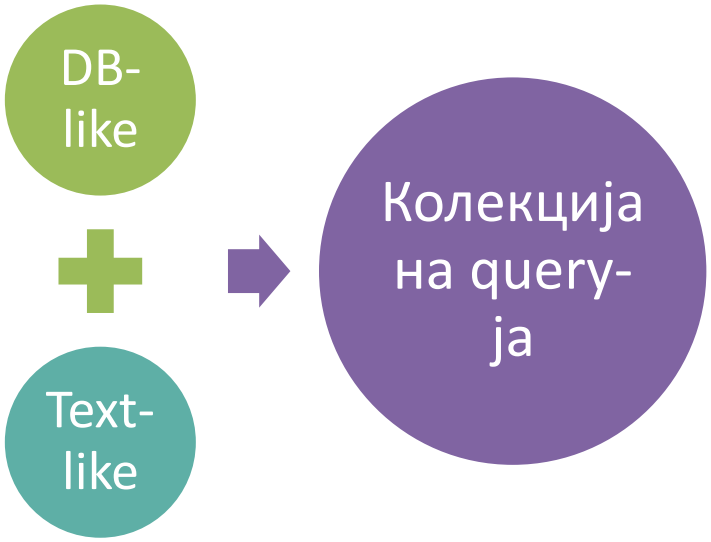
Гореопишната колекција на документи потоа се користи за пребарување и евалуација на добиените резултати и целосната ефективност и ефикасност на системот.

6.1.1.2 Опис на Тест множество од информациски потреби (Queries)

Тест множеството од информациски потреби претставува збир од различни видови на пребарувања (queries) кои се генерирани од експерти, односно лица кои секојдневно ги креираат или пак ги употребуваат на било каков начин документите од тип Тикет. Пребарувањата, т.е. упитите се генерирани врз основа на реални потреби за пронаоѓање на документи во даден момент.

Колекцијата содржи 25 упити кои се поделени во две категории. Иницијално немаше поделеност на упитите во категории. По првата итерација на мерења и резултати беше заклучено дека за одредени упити подобро функционира Артисофт Пребарувачката Машина, додека за останатите подобро функционира традиционалното пребарување во АртАИИС. Врз основа на тоа ги поделивме упитите во две категории кои ќе бидат

опишани во продолжение (Слика 5.29.). Дополнително, и самите резултати се опишани за двете категории индивидуално, како и споредба на двата система за двете категории.



Слика 5.29. Категории на пребарување-ја

Категориите на упити се следните:

1. *Филтер, DB (Database)-ориентирани упити* - Упитите кои припаѓаат на оваа категорија може релативно лесно и точно да генерираат резултати од пребарувањето користејќи ги постоечките филтри за пребарување во ЕРП системот АртАИИС. Самиот тикет е збир од полиња кои се пополнуваат според номенклатури, односно еден вид на формулар, како и од полиња кои се текстуални. Доколку за пребарувањето (упитот) е доволно пополнување на одредени филтер полиња или пак датумски полиња, тогаш упитот припаѓа во оваа категорија. Оваа категорија содржи 14 од вкупно 25 упити.
2. *Текст-ориентирани упити* - Упитите кои припаѓаат на оваа категорија, односно резултатите кои се очекуваат од истите не може да се добијат користејќи ги само постоечките филтри за пребарување во ЕРП системот АртАИИС. Системот или не поседува можност за пребарување според одредени услови или пак генерира огромна листа од резултати која не е подредена според никаква релевантност во однос на она што се пребарува. Овие упити најчесто се комбинација од зборови (зборови) кои може да се појават како дел од било кое од текстуалните, т.е. описните полиња на тикетот. Дополнително во оваа категорија се и оние упити кои можеби се еден вид на меѓу-категорија. Пример

за таков упит е оној кој е комбинација од неколку текстуални зборови и датум или пак дел од датум. Оваа категорија содржи 11 од вкупно 25 упити.

Секој од 25-те упити има повеќе атрибути – идетификационен број, пребарувачки стринг (она што се впишува во полето *Барај* при пребарување), опис на информацискатa потреба, услови под кои еден документ е релевантен за дадениот упит, како и категорија на која истиот припаѓа. Упитите се евидентираат во XML нотација:

Изворен код 5.13. XML репрезентација на упит

```
<queries>
        <query topic="1">
                <search_query> …</search_query>
                 <desc>…</desc>
                 <narrative>…</narrative>
                <query_category> …</query_category>
        </query>
        …
</queries>
```

Дополнително, за секој упит се води и колекција на документи за евалуација, како и вектор на релевантност на оваа колекција за соодветниот упит. Овие податоци ќе бидат опишани во следната секција.

Во продолжение е дадена XML репрезентација на три упити. Упитот со реден број 1 е од категорија DB, додека останатите два од категорија TEXT. Третиот упит е пример за кој припаѓа на одредена меѓу-категорија поради информацискатa потреба од припадност во одреден датумски опсег. Останатите упити се детално прикажани во Прилог 1.

Изворен код 5.14. XML репрезентација на 3 пример упити

```
<?xml version="1.0" encoding="UTF-8"?>
<queries>
    <пребарување topic="1">
        <search_пребарување>македонија сообраќај горан николов</search_пребарување>
        <desc>
                Сите тикети за Македонија Сообраќај по кои работел Горан Николов.
        </desc>
        <narrative>
                Документот е релевантен (се означува со 1) доколку е од тип Тикет за
                коминтентот Македонија Сообраќај. Дополнително тикетот е релевантен само
                доколку лицето Горан Николов на било кој начин е вклучено во обработката
                на тикетот.
        </narrative>
        <пребарување_category>DB</пребарување_category>
    </пребарување>
    <пребарување topic="2">
```

```
        <search_пребарување>неуспешно порамнување исек</search_пребарување>
        <desc>
                Тикети за продукт ИСЕК кои се однесуваат на неупешно порамнување.
        </desc>
        <narrative>
                Документот е релевантен (се означува со 1) доколку е од тип Тикет за
                продукт ИСЕК. Дополнително тикетот е релевантен само доколку се работи за
                пријава, т.е. разрешување на неуспешно порамнување.
        </narrative>
        <пребарување_category>TEXT</пребарување_category>
    </пребарување>
    <пребарување topic="3">
        <search_пребарување>11/2013 спорна неуспешна трансакција</search_пребарување>
        <desc>
                Тикети за продуктот ИСЕК кои се однесуваат на спорни или пак неуспешни
                трансакции во ноември 2013 година.
        </desc>
        <narrative>
                Документот е релевантен (се означува со 1) доколку е од тип Тикет за
                продукт ИСЕК. Датумот на пријава или пак разрешување мора да биде во
                ноември 2013 година. Дополнително тикетот е релевантен само доколку се
                работи за пријава, т.е. разрешување на неуспешна или пак спорна
                трансакција.
        </narrative>
        <пребарување_category>TEXT</пребарување_category>
    </пребарување>
</queries>
```

6.1.1.3 Множество на релевантни документи

Последниот сегмент од тест колекцијата е Множеството на релевантни документи. Резултатите од пребарувањата во двата система, како и листата на релевантни документи потоа се користи како влез во систем кој пресметува одредено множество на мерки кои се користат за оценување и споредба на двата система. Ова е предмет на дискусија во следната секција.

Во оваа секција ќе се задржиме на методологијата според која го формираме множеството на релевантни документи, односно векторот на релевантност за секој упит. Секако, претпоставуваме дека индексот е веќе наполнет со претходно опишаната колекција на документи. Итеративно се изминуваат сите упити и за секое се извршуваат следните чекори кои резултираат со добивање на множество од максимум 80 документи за евалуација за секој упит, како и вектор на релевантност за секој од овие 80 документи:

1. *Пребарување во Пребарувачка Машина* - Се врши пребарување во Артисофт

Пребарувачката Машина со одредено множество на клучни зборови и се евидентира времето на пребарување, како и листа од идентификациони броеви на документите кои се добиваат како резултат сортирани според алгоритам за Scoring на документи на Lucene (OkapiBM25). Редоследот на приказ на даден документ е всушност неговиот ранг. Колку е помал овој број, толку рангот е поголем (документот е меѓу првите резултати).

2. *Пребарување во АртАИИС*- Се врши пребарување во АртАИИС со одредено множество на филтри кои ги дефинира самиот корисник од информацискатa потреба на упитот. Се евидентира времето на пребарување, како и листа од идентификациони броеви на документите кои се добиваат како резултат сортирани според редослед на внес на документите во системот. Редоследот на приказ на даден документ е всушност неговиот ранг. Колку е помал овој број, толку рангот е поголем (документот е меѓу првите резултати).
3. *Множество на резултати кои се повторуваат* – Од евидентираната листа на идентификациони броеви на тикети, како и нивните рангови, програмски се селектираат оние кои се повторуваат како резултат во пребарувањата извршени во првите два чекора. Доколку има 80 или пак помалку од 80 вакви документи сите стануваат дел од множеството на документи за евалуација на релевантност. Доколку има повеќе од 80 вакви документи, се селектираат првите 80 сортирани според средната вредност на рангот на документот во двата система. Во овој случај се прескокнува чекорот 4.
4. *Множество на резултати кои не се повторуваат* – Доколку во претходниот чекор не се формира колекција од 80 документи, истите се дополнуваат со документи кои се појавуваат како резултат само во еден од системите. Распределбата е подеднаква. Имено, доколку треба да се дополнат уште 20 документи, се селектираат 10-те прворангирани документи од двата система кои не се резултат на пребарувањето во двата система, односно оние кои досега не се во колекцијата на документи за евалуација на релевантност.
5. *Оценување на релевантност на упити* – Во овој чекор се избираат експерти, односно лица кои ја познаваат проблематиката и кои ќе ја оценуваат релевантноста на секој од документите во колекцијата на документи за евалуација за дадениот упит. Секој од документите бинарно се означува дали е релевантен за моменталниот упит (со 1 или 0). Оваа листа на вредности го формира векторот на релевантност. Оттука го добиваме и бројот на релевантни

документи за дадениот упит.

6. *Генерирање на влез во Trac* – Од генерираните листи на резултати, придружени со нивните рангови, како и колекцијата на документи за евалуација и векторот на релевантност програмски се генерираат txt документи кои потоа претставуваат влезни аргументи во програма за евалуација која подетално ќе биде опишана во следната секција. При ваквата обработка на резултатите се добиваат и одредени статистики за тест колекцијата.

Во продолжение се прикажани колекцијата на документи за евалуација, векторот на релевантност, како и бројот на релевантни резултати за претходно прикажаните упити во XML нотација.

Изворен код 5.15. XML репрезентација на резултати за 3 пример упити

```
<?xml version="1.0" encoding="UTF-8"?>
<queries>
    <пребарување topic="1">
        <document_evaluation_pool>
                498, 1180, 2648, 829, 305, 465, 830, 662, 306, 1011, 1041, 656, 820, 301,
                822, 496, 585, 495, 930, 888, 300, 821, 721, 2898, 304, 657, 769, 254,
                1434, 2641, 103, 275, 1503, 584, 145, 823, 2483, 1362, 744, 1363, 2649,
                666, 1884, 665, 331, 816, 1464, 329, 322, 771, 1768, 2110, 2650, 174, 82,
                590, 36, 100, 754, 1026, 618, 1788, 2158, 271, 494, 1308, 307, 52, 84, 83,
                499, 1399, 37, 522, 166, 2157, 582, 1300, 583, 521
        </document_evaluation_pool>
        <document_relevance_vector>
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1
        </document_relevance_vector>
        <no_relevant_documents>80</no_relevant_documents>
    </пребарување>
    <пребарување topic="2">
        <document_evaluation_pool>
                348, 14, 42, 16, 49, 327, 2477, 175, 26, 40, 17, 7, 272, 154, 30, 1, 212,
                47, 1659, 426, 15, 1713, 106, 357, 11, 133, 1344, 1657, 2418, 21, 31, 20,
                134, 9, 278, 206, 25, 24, 1277, 387, 43, 147, 45, 44, 23, 38, 6, 39, 41,
                311, 356, 336, 32, 12, 2467, 159, 2163, 19, 182, 13, 48, 286, 46, 50, 126,
                10, 2660, 994, 8, 22, 1191, 18, 299, 116, 381, 2215, 34, 5, 296, 27
        </document_evaluation_pool>
        <document_relevance_vector>
                1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0,
                1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
                1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1,
                1, 0, 0, 1, 0
        </document_relevance_vector>
        <no_relevant_documents>42</no_relevant_documents>
    </пребарување>
```

```
    <пребарување topic="3">
        <document_evaluation_pool>
                2440, 2432, 2500, 2514, 2587, 2444, 2422, 2467, 2456, 2468, 2420, 2470,
                2532, 2525, 2490, 2474, 2473, 2600, 2492, 2458, 2434, 2550, 2438, 2595,
                2477, 2577, 2556, 2528, 2471, 2584, 2594, 2480, 2478, 2560, 2572, 2576,
                2460, 2531, 2501, 2421, 2578, 2509, 2605, 2499, 2483, 2429, 2469, 2575,
                2511, 2606, 2547, 2423, 2512, 2475, 2565, 2513, 2481, 2559, 2428, 2486,
                2583, 2463, 2479, 2426, 2424, 2537, 2427, 2582, 2472, 2536, 2526, 2439,
                2493, 2564, 2551, 2482, 2465, 2461, 2464, 2476
        </document_evaluation_pool>
        <document_relevance_vector>
                0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
                0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0,
                0, 0, 1, 1, 0
        </document_relevance_vector>
        <no_relevant_documents>19</no_relevant_documents>
    </пребарување>
</queries>
```

Во продолжение се дадени табели во кои се прикажани минималниот, максималниот и просечниот број на резултати во двата система, како и минималниот, максималниот и просечниот број на релевантни резултати во колекцијата на документи за евалуација на релевантност, најпрво сумарно, а потоа за двете категории на упити индивидуално.

Табела 5.2. Статистички податоци за тест колекцијата за двете категории

	Минимален број	**Максимален број**	**Просечен број**
Пребарувачка Машина	2899	22	1474.4
АртАИИС	2445	0	458.32
Релевантни док. во колекција на док. за евалуација	80	0	39.88

Табела 5.3. Статистички податоци за тест колекцијата за категорија TEXT

	Минимален број	**Максимален број**	**Просечен број**
Пребарувачка Машина	2899	22	1474.4
АртАИИС	2445	0	458.32
Релевантни док. во колекција на док. за евалуација	80	0	39.88

Табела 5.4. Статистички податоци за тест колекцијата за категорија DB

	Минимален број	**Максимален број**	**Просечен број**
Пребарувачка Машина	2894	25	1394
АртАИИС	2445	0	659.54
Релевантни док. во колекција на док. за евалуација	80	7	28.72

6.1.2 Тестирање и експериментални резултати

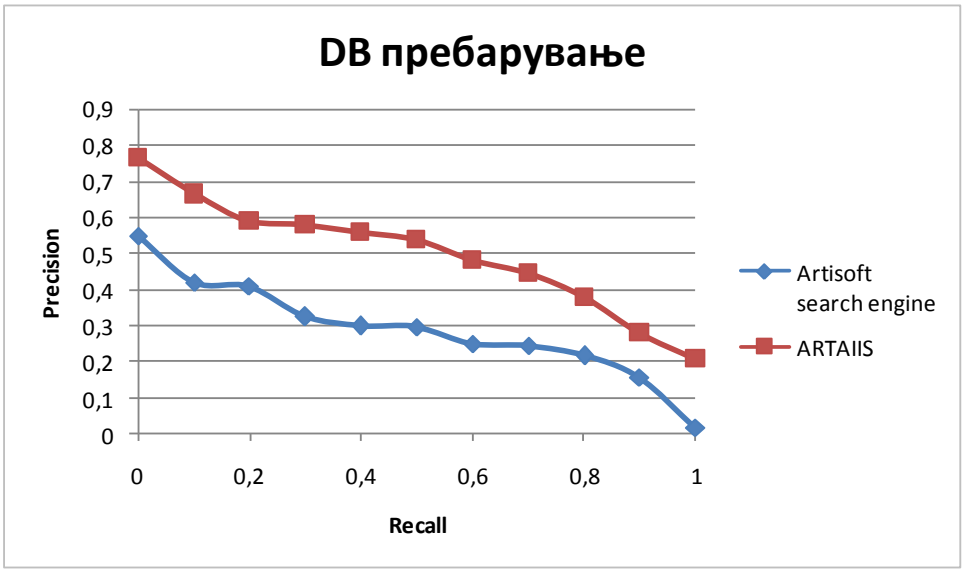
Во продолжение се прикажани резултатите, табеларно (Табела 5.5 и Табела 5.6) и графички (Слика 6.1 и Слика 6.2) од евалуација на двата системи за пребарување со двете категории, DB базирани и Текст базирани упити.

Табела 5.5. Резултати од пребарување со DB базирани упити

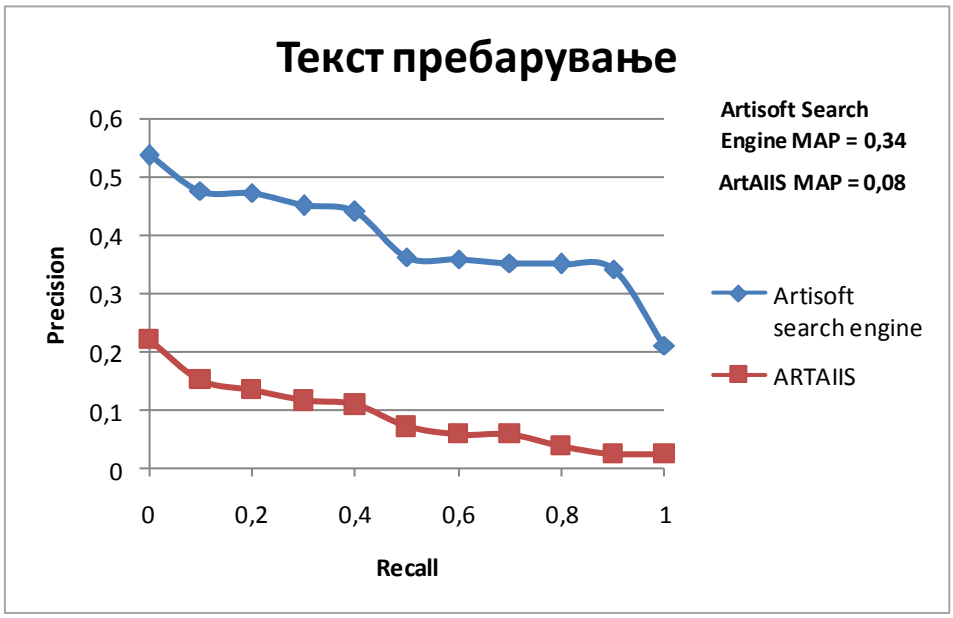
DB БАЗИРАНО			
Артисофт пребарувачка машина		**АртАИИС**	
Метрика	Вредност	**Метрика**	Вредност
P5	0.27	**P5**	0.43
P10	0.26	**P10**	0.44
P15	0.25	**P15**	0.42
P20	0.26	**P20**	0.42
P30	0.27	**P30**	0.42
P100	0.28	**P100**	0.41
MAP	0.25	**MAP**	0.45
AP	0.05	**AP**	0.19

Табела 5.6. Резултати од пребарување со Текст базирани упити

ТЕКСТ БАЗИРАНО			
Артисофт пребарувачка машина		**АртАИИС**	
Метрика	Вредност	**Метрика**	Вредност
P5	0.36	**P5**	0.11
P10	0.32	**P10**	0.07
P15	0.31	**P15**	0.08
P20	0.30	**P20**	0.08
P30	0.25	**P30**	0.08
P100	0.27	**P100**	0.08
MAP	0.34	**MAP**	0.08
AP	0.22	**AP**	0.01



Слика 6.1. Резултати Precision и Recall од пребарување со DB базирани упити



Слика 6.2. Резултати Precision и Recall од пребарување со Текст базирани упити

7 Заклучок и идна работа

Во последната секција од овој труд се презентирани заклучоците и дискусиите за резултатите од направената евалуација на развиената пребарувачка машина. На крај даден е осврт на можните идни истражувања и насоки во кои би се движеле оптимизацијата и надградбата на Артисофт Пребарувачката Машина во рамки на АртАИИС ЕРП системот и други типови на компаниски ИС.

7.1 Заклучни согледувања

Резултатите од направената евалуација на двата типа на пребарување во ЕРП системот АртАИИС беа прикажани преку стандардни мерки за мерење на ефективноста на ваквите пребарувачки системи.

Резултатите се однесуваа на две различни категории на упити во ЕРП системот АртАИИС, DB и Текст базирани. Во случајот на DB базираните упити пребарувањето преку филтри се покажаа подобро од Текст базираното пребарување (Слика 6.1), што беше и очекувано со оглед дека во овој случај филтер базираното пребарување е потесно дефинирано и со полиња кои постојат како атрибути во базата. На ова особено укажува вредноста на MAP во двата случаи прикажани во Табела 5.5. Но треба да се напомене дека временските разлики не се значајни споредени со вкупното време потребно за пребарување.

За разлика од нив, резултатите од упитите од Текст базираната категорија покажаа дека Артисофт Пребарувачката Машина е поефикасна во пребарувањето документи во секој случај (Слика 6.2).

Од направените анализи може да се заклучи дека двата типа на пребарување може да бидат користени за пребарување информации низ ЕРП системот. Она што е позначајно и во нашиот случај релевантно е дека пребарувањата со АПМ се ефикасни и брзи и во двата случаи, иако во пребарувањето на податоци низ табели со филтерско пребарување симболично побрзо се доаѓа до резултат (не теба да се заборави дека тука не е внесено времето на доаѓање до екранот за пребарување како и внесување на сите параметри за пребарување и конечно листата на вратени документи не е подредена по релевантност па и тоа додава време кое треба да се вкалкулира). Истотака мора да се истакне дека едноставен внес на зборови во рамка на пребарувачка машина достапна насекаде низ ЕРП системот ќе придонесе да се зголеми количеството на информации барани од системот (едноставноста ќе ги натера корисниците сами да доаѓаат до

информации без да одат да ги бараат од искусните колеги) а со то ќе се намали вкупното време кое се троши за доаѓање до потребните информации како и ке се зголеми вкупниот број на побарани информации.

7.2 Идни истражувања

Во иднина се планирани неколку дополнителни функционалности и оптимизации на Артисофт Пребарувачката Машина во поглед на претпроцесирањето на документите како и во алгоритамот за рангирање на документите. Примена на неколку стандардни техники за претпроцесирање на документите би го подобриле времето на вратен одговор до корисникот во случај на постоечката и поголема колекција на документи, и би го подобриле враќањето на порелевантни документи за корисникот.

Дополнително поддршка на латинично и кирилично пребарување е истотака потребна идна надградба на пребарувачката машина со цел да се овозможи намалено време за доаѓање до потребната информација преку олеснување на корисничката комуникација со системот.

Вака развиената пребарувачка машина во иднина би можела да се евалуира и над други ЕРП системи па и повеќе неинтегрирани компаниски ИС на поголема документ колекција каде и дополнително би се искористила и статистичката мерка t-тест.

8 Библиографија

1. Anotny, T, (2013). From Keywords to Contexts. The MOZ Blog http://moz.com/blog/from-keywords-to-contexts-the-new-пребарување-model
2. Baeza-Yates, R, Ribeiro-Neto, B, (1999). *Modern Information Retrieval.* First Edition, Addison Wesley.
3. Baeza-Yates, R, Ribeiro-Neto, B, (2010). *Modern Information Retrieval.* Second Edition, Addison Wesley.
   http://grupoweb.upf.es/WRG/mir2ed/pdf/slides_chap15.pdf
   Buttcher, S, Clarke, C, L, A, Cormack, G, V, (2010). Information Retrieval: Implementing and Evaluating Search Engines. MIT Press
   http://www.ir.uwaterloo.ca/book/
4. Chang, S, I, Gable, G, Smythe, E, Timbrell, G, (2000). A Delphi examination of public sector ERP implementation issues. In *Proceedings 21st International Conference on Information Systems,* pp. 494-500, Brisbane, Qld.
   http://eprints.qut.edu.au/4746/1/4746.pdf
5. Frey, L, (2013). Reinventing the ERP Engine. Tech Trends 2013 – Enablers
   http://www.deloitte.com/assets/Dcom-UnitedKingdom/Local%20Assets/Documents/Services/Consulting/uk-con-reinventing-erp-engine.pdf
6. Geibelmann, B, (2010). *Information Retrieval Systems on the Internet Seminar Information Retrieval*. University of Konstanz
7. González, R, B, (2008). *Index Compression for Information Retrieval Systems.* Ph.D. THESIS, University of Coruna http://www.dc.fi.udc.es/~roi/publications/rblanco-phd.pdf
8. Grossman, D, A, (2004). Information Retrieval: Algorithms and Heuristics (The Information Retrieval Series), Springer; 2nd edition, ISBN-13: 978-1402030048
   http://www.amazon.com/Information-Retrieval-Algorithms-Heuristics-Edition/dp/1402030045
9. Halim, N. 2013, Stream processing. Director and Chief Architect of Big Data, IBM.
   http://www.ibm.com/smarter-computing/us/en/technical-breakthroughs/stream-processing.html
10. Hawkes, R, (2013). What is Enterprise Search?, RAVN, Power of Understanding, Blog. Објавено на 20.03.2013. Прочитано на 15.11.2013

http://www.ravn.co.uk/2013/03/20/enterprise-search/?goback=.gde_1778219_member_225014424

11. Hiemstra, D, (2009). *Information Retrieval Models.* Searching in the 21st Century. John Wiley and Sons, Ltd., ISBN-13: 978-0470027622 http://wwwhome.cs.utwente.nl/~hiemstra/papers/IRModelsTutorial-draft.pdf
12. Inkpen, D, (2013). *Information Retrieval on the Internet*, CSI4107: Information Retrieval and the Internet, University of Ottawa http://www.site.uottawa.ca/~diana/csi4107/IR_draft.pdf
13. Langville, N, A, Meyer, C, D, (2006). *Information Retrieval and Web Search*, The Handbook of Linear Algebra. CRC Press http://meyer.math.ncsu.edu/Meyer/PS_Files/HLA.pdf
14. Sun, L, (2009). Smart Search Engine for Information retrieval, Masters thesis, Durham University http://etheses.dur.ac.uk/72/
15. Manning, C, D, Raghavan, P, Schütze, H, (2009). *An Introduction to Information Retrieval*, Cambridge University Press
16. Matthews, D, (2013). *Why enterprise application search is crucial to your ERP system?* White paper, IFSWorld. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&ved=0CEIQFjAA&url=http%3A%2F%2Fwww.ifsworld.com%2Fsi-lk%2Fdownload%2Fresources%2Fpdfs%2Fwhite-papers%2Fwhy-enterprise-application-search-eas-is-crucial-to-your-erp-system%2Fwhite-paper-why-enterprise-application-search-is-crucial-to-your-erp-system.pdf&ei=7TSXUs2yEIOqtAbVrIDwCA&usg=AFQjCNEiChJs6jWWHOTKmYGqaDtGTqre_A&sig2=eYX0GMIotOGDFvj5mUlz1Q&bvm=bv.57155469,d.Yms Mihaescu, C, (2009). Algorithms for Information Retrieval – Introduction. Laboratory exercisess, Universitatea din Craiova http://software.ucv.ro/~cmihaescu/ro/teaching/AIR/docs/Lab1-Algorithms%20for%20Information%20Retrieval.%20Introduction.pdf
17. O'Brien, James (2011). *Management Information Systems(MIS)*. New York: McGraw-Hill, Irwin. p. 324.
18. Okuttah, M, (2013). Google *launches search tool to help firms retrieve data.* BusinessDaily.

http://www.businessdailyafrica.com/Corporate-News/Google-launches-search-tool-to-help-firms-retrieve-data/-/539550/1925420/-/5kn5dnz/-/index.html

19. Salton, G, Wong, A, Yang, C, S, (1975). *A vector space model for automatic indexing.* Communications of the ACM, 18(11):613–620
20. Sheilds, M, G, (2001). *E-Business and ERP: Rapid Implementation and Project Planning.* John Wiley and Sons, Inc. p. 9-10.
21. Sheilds, M, G, (2005). *E-Business and ERP: Rapid Implementation and Project Planning.* John Wiley and Sons, Inc. p. 9.
22. Wills, R, S, (2006). Google's PageRank: The Math Behind the Search Engine, Department of Mathematics, North Carolina State University http://www.cems.uvm.edu/~tlakoba/AppliedUGMath/other_Google/Wills.pdf
23. Witten, I, H, Moffat, A, Bell, T, C, (1999). *Managing Gigabytes: Compressing and Indexing Documents and Images, Second Edition.* Morgan Kaufmann Publishers.
24. Ziliak, P, (2013). Powerful ERP search tool now available for Sage 100 ERP v2013. http://www.prweb.com/releases/2013/2/prweb10478394.htm, прегледано на 10.12.2013
25. Zobel, J, Moffat, A, (1998). *Exploring the similarity space.* ACM SIGIR Forum, 32(1):18–34
26. Enterprise Search, Modern Information Retrieval, Addison Wesley, 2010

9 Прилог 1


```
<?xml version="1.0" encoding="UTF-8" ?>
- <queries>
- <пребарување topic="1">
  <search_пребарување>македонија сообраќај се чека од клиент 2013</search_пребарување>
  <desc>Тикети од Македонија Сообраќај кои се пријавени во 2013 година, а имаат или пак во текот на животниот циклус на документот имале статус се чека на одговор од клиент.</desc>
  <narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет за коминтентот Македонија Сообраќај. Датумот на пријава или пак разрешување мора да биде во 2013 година. Дополнително тикетот е релевантен само доколку истиот во даден момент имал статус се чека на одговор од клиент.</narrative>
  <пребарување_category>DB</пребарување_category>
  <document_evaluation_pool>98, 830, 666, 665, 851, 634, 2, 160, 1300, 771, 584, 2641, 495, 829, 656, 465, 305, 69, 823, 1201, 1007, 630, 1363, 512, 71, 985, 265, 2676, 721, 522, 1284, 1286, 271, 1014, 821, 2677, 280, 1366, 2689, 1041, 816, 1011, 657, 167, 1600, 585, 1393, 1464, 781, 499, 1203, 158, 820, 517, 1362, 1294, 780, 1177, 837, 810, 848, 590, 1573, 1261, 1392, 304, 662, 2447, 722, 300, 2463, 194, 992, 963, 322, 383, 2527, 597, 849, 2898</document_evaluation_pool>
  <document_relevance_vector>0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0</document_relevance_vector>
  <no_relevant_documents>7</no_relevant_documents>
  </пребарување>
- <пребарување topic="2">
  <search_пребарување>АртАИИС настана грешка при внес во табелите</search_пребарување>
  <desc>Тикети за продукт АртАИИС во кои се појавува грешката Настана грешка при внес во табелите во било кое од описните полиња на тикетот.</desc>
  <narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет за продуктот АртАИИС. Дополнително тикетот е релевантен само доколку во некое од описните полиња на истиот се појавува содржината Настана
```

грешка при внес во табелите.</narrative>
<пребарување_category>TEXT</пребарување_category>
<document_evaluation_pool>733, 369, 845, 522, 987, 129, 70, 275, 846, 664, 305, 857, 944, 67, 28, 862, 950, 718, 82, 300, 301, 35, 103, 1075, 145, 84, 331, 499, 967, 818, 589, 1145, 948, 810, 271, 1408, 656, 744, 72, 954, 814, 1148, 199, 174, 719, 717, 700, 1087, 1107, 1294, 52, 952, 69, 83, 265, 949, 29, 521, 716, 811, 65, 71, 580, 512, 1211, 37, 699, 1051, 825, 1302, 720, 498, 329, 460, 888, 826, 33, 590, 951, 1600</document_evaluation_pool>
<document_relevance_vector>0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1</document_relevance_vector>
<no_relevant_documents>7</no_relevant_documents>
</пребарување>
- <пребарување topic="3">
<search_пребарување>АЛМА-М фактури</search_пребарување>
<desc>Тикети за коминтент АЛМА-М, а кои се однесуваат на проблем со фактури.</desc>
<narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет за коминтентот АЛМА-М. Дополнително тикетот е релевантен само доколку истиот се однесува на проблем со процесирање, генерирање или обработка на фактури.</narrative>
<пребарување_category>DB</пребарување_category>
<document_evaluation_pool>246, 89, 43, 10, 73, 62, 15, 23, 7, 230, 56, 5, 42, 187, 50, 34, 54, 374, 77, 79, 76, 44, 86, 22, 88, 30, 32, 63, 549, 87, 38, 74, 40, 81, 20, 80, 151, 31, 58, 55, 47, 21, 59, 595, 91, 25, 61, 49, 24, 57, 17, 48, 27, 78, 302, 14, 60, 1, 64, 75, 154, 157, 6, 51, 39, 90, 26, 8, 573, 46, 9, 45, 41, 12, 142, 16, 85, 11, 18, 13</document_evaluation_pool>
<document_relevance_vector>0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0</document_relevance_vector>
<no_relevant_documents>0</no_relevant_documents>
</пребарување>
- <пребарување topic="4">
<search_пребарување>македонија сообраќај надица мицоска јован милисов 2012</search_пребарување>

<desc>**Тикети пријавени од лице Надица Мицоска, а по кои работел Јован Милисов во 2012 година.**</desc>
<narrative>**Документот е релевантен (се означува со 1) доколку е од тип Тикет за коминтентот Македонија Сообраќај. Датумот на пријава или пак разрешување мора да биде во 2012 година. Дополнително тикетот е релевантен само доколку е пријавен од лицето Надица Мицоска, а по истиот работело лицето Јован Милисов.**</narrative>
<пребарување_category>**DB**</пребарување_category>
<document_evaluation_pool>**1046, 2154, 1396, 2002, 1611, 517, 1216, 2503, 1887, 1398, 1284, 1014, 1391, 2595, 1141, 2156, 460, 2527, 2596, 2675, 2594, 2768, 1886, 1996, 2790, 2727, 2687, 1883, 1393, 1294, 2676, 1919, 1298, 2725, 1285, 1432, 1395, 1198, 992, 1903, 696, 1600, 2677, 664, 2155, 1997, 1287, 2159, 1430, 2664, 1542, 2597, 495, 2543, 1248, 2646, 2177, 1399, 1286, 2726, 1608, 2533, 695, 1004, 1397, 174, 663, 2222, 899, 683, 2458, 694, 2534, 697, 2457, 1912, 845, 1302, 1392, 1888**</document_evaluation_pool>
<document_relevance_vector>**0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0**</document_relevance_vector>
<no_relevant_documents>**31**</no_relevant_documents>
</пребарување>
- <пребарување topic="**5**">
<search_пребарување>**македонија сообраќај висок приоритет**</search_пребарување>
<desc>**Тикети за Македонија Сообраќај со висок приоритет.**</desc>
<narrative>**Документот е релевантен (се означува со 1) доколку е од тип Тикет за коминтентот Македонија Сообраќај. Дополнително тикетот е релевантен само доколку истиот има висок приоритет за клиентот.**</narrative>
<пребарување_category>**DB**</пребарување_category>
<document_evaluation_pool>**2619, 33, 2135, 1041, 2002, 460, 2311, 29, 1969, 2156, 2354, 1664, 2331, 1298, 1216, 2852, 1248, 1302, 265, 785, 2693, 2623, 2898, 174, 1611, 899, 1294, 2853, 967, 305, 683, 944, 1087, 2400, 1430, 1450, 2527, 1282, 300, 992, 1751, 2825, 271, 52, 82, 2332, 307, 1927, 2480, 28, 888, 1608, 301, 889, 1600, 909, 2333, 2329, 465, 2089, 517, 37, 83, 1177, 2646, 2340, 1573, 1045, 1046, 780, 1292, 1141, 326, 1542, 1288, 1464, 145, 776, 1380, 2155**</document_evaluation_pool>
<document_relevance_vector>**1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,**

0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0</document_relevance_vector>
<no_relevant_documents>20</no_relevant_documents>

</пребарување>

- <пребарување topic="6">

<search_пребарување>Business барање дополнување</search_пребарување>

<desc>Тикети со ниво на услуга Business во кои се бара некаков вид на дополнување.</desc>

<narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет за ниво на услуга Business. Дополнително тикетот е релевантен доколку во истиот се бара дополнување на дадена функционалност.</narrative>

<пребарување_category>DB</пребарување_category>

<document_evaluation_pool>1087, 699, 2904, 2596, 1045, 1362, 2177, 3, 2193, 2185, 271, 2458, 1919, 1177, 1106, 404, 1287, 2153, 1206, 1308, 83, 1398, 2402, 322, 1703, 2677, 1888, 657, 16, 2676, 1299, 2675, 82, 1964, 845, 2678, 992, 12, 696, 4, 329, 2534, 1886, 68, 103, 72, 1762, 36, 695, 2176, 460, 268, 1777, 830, 2646, 2222, 98, 780, 292, 1285, 1393, 383, 959, 65, 864, 2781, 889, 1430, 663, 929, 2696, 21, 1887, 69, 15, 2726, 1011, 1730, 2866, 1996</document_evaluation_pool>

<document_relevance_vector>1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1</document_relevance_vector>
<no_relevant_documents>68</no_relevant_documents>

</пребарување>

- <пребарување topic="7">

<search_пребарување>множествоирање ПОС телефон 11/2012</search_пребарување>

<desc>Тикети кои се однесуваат на множествоирање ПОС зборинал преку телефон во ноември 2012 година.</desc>

<narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет и датумот на пријава или пак разрешување мора да биде во ноември 2012 година. Дополнително тикетот е релевантен само доколку се однесува на помош при множествоирање на ПОС зборинал преку телефон.</narrative>

<пребарување_category>TEXT</пребарување_category>

<document_evaluation_pool>112, 106, 13, 2, 95, 185, 7, 44, 31, 26, 14, 16, 35, 22,

33, 53, 136, 20, 3, 12, 1, 23, 21, 24, 114, 32, 30, 6, 40, 48, 143, 75, 37, 110, 104, 82, 160, 4, 15, 38, 9, 42, 189, 36, 28, 18, 191, 19, 55, 117, 60, 107, 180, 39, 10, 49, 41, 62, 63, 64, 61, 94, 17, 29, 25, 11, 8, 80, 27, 184, 59, 83, 98, 50, 125, 87, 5, 45, 43, 86</document_evaluation_pool>

<document_relevance_vector>1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1</document_relevance_vector>

<no_relevant_documents>19</no_relevant_documents>

</пребарување>

- <пребарување topic="8">

<search_пребарување>11/2013</search_пребарување>

<desc>Тикети кои се отворени во 11/2013.</desc>

<narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет и датумот на пријава или пак разрешување мора да биде во ноември 2013 година.</narrative>

<пребарување_category>DB</пребарување_category>

<document_evaluation_pool>2492, 2424, 2499, 2422, 2470, 2476, 2434, 2501, 2468, 2572, 2426, 2531, 2473, 2526, 2471, 2584, 2578, 2472, 2550, 2420, 2467, 2525, 2560, 2500, 2490, 2423, 2582, 2605, 2564, 2583, 2536, 2514, 2481, 2438, 2556, 2475, 2509, 2428, 2575, 2482, 2456, 2512, 2469, 2478, 2474, 2547, 2511, 2421, 2576, 2439, 2565, 2594, 2465, 2532, 2537, 2559, 2479, 2513, 2587, 2477, 2440, 2595, 2429, 2600, 2432, 2460, 2464, 2551, 2493, 2458, 2483, 2463, 2431, 2486, 2480, 2461, 2444, 2427, 2577, 2430</document_evaluation_pool>

<document_relevance_vector>1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1</document_relevance_vector>

<no_relevant_documents>80</no_relevant_documents>

</пребарување>

- <пребарување topic="9">

<search_пребарување>грешка македонија сообраќај</search_пребарување>

<desc>Тикети кои го содржат зборот грешка во информацијата до одговорно лице за коминтент Македонија Сообраќај.</desc>

<narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет за коминтентот Македонија Сообраќај. Дополнително тикетот е релевантен

само доколку во информацијата до одговорно лице се појавува зборот грешка.</narrative>
<пребарување_category>TEXT</пребарување_category>
<document_evaluation_pool>52, 846, 837, 495, 29, 771, 722, 499, 460, 657, 697, 100, 174, 662, 65, 37, 67, 584, 944, 780, 820, 83, 1302, 744, 630, 129, 930, 666, 275, 821, 33, 816, 694, 4, 265, 753, 776, 781, 82, 590, 1768, 512, 585, 829, 70, 145, 778, 71, 28, 1041, 465, 696, 36, 271, 35, 1600, 305, 69, 307, 830, 634, 683, 326, 772, 68, 301, 84, 663, 851, 665, 823, 300, 695, 664, 72, 721, 66, 656, 98, 517</document_evaluation_pool>
<document_relevance_vector>0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0</document_relevance_vector>
<no_relevant_documents>15</no_relevant_documents>
</пребарување>
- <пребарување topic="10">
<search_пребарување>модификација промена 2012</search_пребарување>
<desc>Тикети од 2012 година кои го содржат зборот модификација/промена во било кое од полињата.</desc>
<narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет, а датумот на пријава или пак разрешување мора да биде во 2012 година. Дополнително тикетот е релевантен само доколку истиот се однесува на било каква модификација во софтверот кој го користи комитентот кој го пријавил тикетот.</narrative>
<пребарување_category>TEXT</пребарување_category>
<document_evaluation_pool>114, 55, 33, 83, 6, 178, 59, 82, 37, 24, 21, 31, 48, 100, 61, 51, 160, 143, 2, 15, 1, 38, 34, 177, 16, 12, 167, 20, 66, 11, 13, 23, 17, 36, 54, 200, 40, 25, 5, 3, 22, 49, 43, 27, 8, 46, 62, 52, 35, 28, 60, 113, 96, 56, 26, 50, 64, 41, 19, 42, 44, 7, 58, 9, 14, 4, 30, 103, 47, 57, 53, 32, 45, 104, 10, 98, 29, 39, 18, 179</document_evaluation_pool>
<document_relevance_vector>0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0</document_relevance_vector>
<no_relevant_documents>8</no_relevant_documents>
</пребарување>
- <пребарување topic="11">

<search_пребарување>**фротирка артикли**</search_пребарување>
<desc>**Тикети кои се однесуваат на комintент Фротирка и го содржат зборот артикли или артикал во било кое од полињата на тикетот.**</desc>
<narrative>**Документот е релевантен (се означува со 1) доколку е од тип Тикет за комintентот Фротирка. Дополнително тикетот е релевантен само доколку истиот го содржи зборот артикал, односно артикли во било кое од описните полиња.**</narrative>
<пребарување_category>**TEXT**</пребарување_category>
<document_evaluation_pool>**1781, 1811, 1146, 1075, 1784, 1614, 2253, 1450, 1145, 1219, 1441, 1148, 2447, 1164, 1408, 2387, 1722, 1107, 1560, 2118, 2298, 1299, 662, 1776, 1593, 1496, 1211, 2029, 1436, 160**</document_evaluation_pool>
<document_relevance_vector>**0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0**</document_relevance_vector>
<no_relevant_documents>**7**</no_relevant_documents>
</пребарување>
- <пребарување topic="**12**">
<search_пребарување>**2012 заглавен зборинал**</search_пребарување>
<desc>**Тикети кои се однесуваат на заглавени зборинали во 2012 година.**</desc>
<narrative>**Документот е релевантен (се означува со 1) доколку е од тип Тикет, а датумот на пријава или пак разрешување мора да биде во 2012 година. Дополнително тикетот е релевантен само доколку истиот се однесува на разрешување на ситуација при заглавување на ПОС зборинал.**</narrative>
<пребарување_category>**TEXT**</пребарување_category>
<document_evaluation_pool>**122, 171, 23, 78, 63, 221, 3, 16, 11, 8, 46, 13, 36, 162, 22, 56, 39, 121, 135, 53, 119, 29, 10, 97, 20, 77, 17, 115, 18, 128, 51, 19, 5, 219, 74, 150, 108, 12, 96, 7, 149, 123, 1, 76, 47, 109, 127, 148, 140, 164, 137, 24, 120, 54, 41, 223, 176, 124, 9, 144, 21, 27, 220, 28, 234, 60, 238, 79, 37, 14, 40, 35, 99, 81, 141, 2, 6, 4, 61, 15**</document_evaluation_pool>
<document_relevance_vector>**1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0**</document_relevance_vector>
<no_relevant_documents>**57**</no_relevant_documents>
</пребарување>
- <пребарување topic="**13**">

<search_пребарување>**11/2013 спорна неуспешна трансакција**</search_пребарување>
<desc>**Тикети за продуктот ИСЕК кои се однесуваат на спорни или пак неуспешни трансакции во ноември 2013 година.**</desc>
<narrative>**Документот е релевантен (се означува со 1) доколку е од тип Тикет за продукт ИСЕК. Датумот на пријава или пак разрешување мора да биде во ноември 2013 година. Дополнително тикетот е релевантен само доколку се работи за пријава, т.е. разрешување на неуспешна или пак спорна трансакција.**</narrative>
<пребарување_category>**TEXT**</пребарување_category>
<document_evaluation_pool>**2440, 2432, 2500, 2514, 2587, 2444, 2422, 2467, 2456, 2468, 2420, 2470, 2532, 2525, 2490, 2474, 2473, 2600, 2492, 2458, 2434, 2550, 2438, 2595, 2477, 2577, 2556, 2528, 2471, 2584, 2594, 2480, 2478, 2560, 2572, 2576, 2460, 2531, 2501, 2421, 2578, 2509, 2605, 2499, 2483, 2429, 2469, 2575, 2511, 2606, 2547, 2423, 2512, 2475, 2565, 2513, 2481, 2559, 2428, 2486, 2583, 2463, 2479, 2426, 2424, 2537, 2427, 2582, 2472, 2536, 2526, 2439, 2493, 2564, 2551, 2482, 2465, 2461, 2464, 2476**</document_evaluation_pool>
<document_relevance_vector>**0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0**</document_relevance_vector>
<no_relevant_documents>**19**</no_relevant_documents>
</пребарување>
- <пребарување topic="**14**">
<search_пребарување>**АртАИИС книжење**</search_пребарување>
<desc>**Тикети за продукт АртАИИС, а го содржат зборот книжење во било кое од полињата.**</desc>
<narrative>**Документот е релевантен (се означува со 1) доколку е од тип Тикет за продуктот АртАИИС. Дополнително тикетот е релевантен само доколку во некое од описните полиња на истиот се појавува зборот книжење.**</narrative>
<пребарување_category>**TEXT**</пребарување_category>
<document_evaluation_pool>**98, 166, 167, 992, 1396, 37, 210, 174, 217, 158, 1398, 216, 29, 1216, 36, 103, 71, 208, 194, 52, 271, 207, 177, 265, 195, 69, 2527, 53, 240, 1996, 209, 1600, 2, 1287, 145, 280, 2664, 275, 2597, 2768, 28, 65, 1284, 252, 498, 199, 100, 35, 84, 279, 82, 2790, 4, 1399, 66, 33, 178, 68, 104, 254,**

160, 1542, 239, 67, 2543, 2291, 301, 300, 2157, 152, 143, 2457, 179, 1198, 72, 70, 517, 129, 83, 3</document_evaluation_pool>
<document_relevance_vector>0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0</document_relevance_vector>
<no_relevant_documents>23</no_relevant_documents>
</пребарување>
- <пребарување topic="15">
<search_пребарување>Хана Ма исек</search_пребарување>
<desc>Тикети за продукт ИСЕК кои се однесуваат на дилерот Хана Ма.</desc>
<narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет за продуктот ИСЕК. Дополнително тикетот е релевантен само доколку се однесува на проблем кај дилерот Хана Ма.</narrative>
<пребарување_category>TEXT</пребарување_category>
<document_evaluation_pool>45, 1567, 19, 26, 1554, 1539, 1553, 1097, 27, 1870, 16, 1, 1092, 15, 34, 8, 22, 44, 24, 46, 31, 1333, 1882, 13, 30, 1151, 1853, 11, 1852, 7, 1698, 1025, 17, 23, 1445, 41, 94, 1060, 2219, 12, 20, 38, 1967, 18, 40, 9, 1552, 47, 48, 1558, 2775, 1615, 1860, 1471, 10, 1859, 21, 1318, 1546, 1970, 1101, 1376, 1622, 1999, 39, 1849, 25, 1601, 32, 42, 6, 2903, 1405, 1255, 1879, 1861, 43, 14, 1260, 5</document_evaluation_pool>
<document_relevance_vector>0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0</document_relevance_vector>
<no_relevant_documents>39</no_relevant_documents>
</пребарување>
- <пребарување topic="16">
<search_пребарување>неуспешно порамнување исек</search_пребарување>
<desc>Тикети за продукт ИСЕК кои се однесуваат на неупешно порамнување.</desc>
<narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет за продукт ИСЕК. Дополнително тикетот е релевантен само доколку се работи за пријава, т.е. разрешување на неуспешно порамнување.</narrative>
<пребарување_category>TEXT</пребарување_category>
<document_evaluation_pool>348, 14, 42, 16, 49, 327, 2477, 175, 26, 40, 17, 7, 272,

154, 30, 1, 212, 47, 1659, 426, 15, 1713, 106, 357, 11, 133, 1344, 1657, 2418, 21, 31, 20, 134, 9, 278, 206, 25, 24, 1277, 387, 43, 147, 45, 44, 23, 38, 6, 39, 41, 311, 356, 336, 32, 12, 2467, 159, 2163, 19, 182, 13, 48, 286, 46, 50, 126, 10, 2660, 994, 8, 22, 1191, 18, 299, 116, 381, 2215, 34, 5, 296, 27</document_evaluation_pool>
<document_relevance_vector>1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0</document_relevance_vector>
<no_relevant_documents>42</no_relevant_documents>
</пребарување>
- <пребарување topic="17">
<search_пребарување>македонија сообраќај</search_пребарување>
<desc>Сите тикети за Македонија Сообраќај.</desc>
<narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет за коминтентот Македонија Сообраќај.</narrative>
<пребарување_category>DB</пребарување_category>
<document_evaluation_pool>305, 4, 584, 33, 512, 697, 307, 944, 695, 69, 776, 72, 657, 70, 66, 846, 82, 987, 103, 67, 300, 820, 837, 930, 129, 992, 664, 585, 1011, 721, 98, 830, 174, 100, 683, 696, 634, 1041, 65, 68, 499, 781, 83, 29, 265, 28, 37, 301, 662, 722, 665, 778, 71, 630, 656, 52, 823, 816, 780, 753, 666, 851, 326, 663, 771, 36, 517, 35, 821, 145, 495, 829, 465, 694, 271, 772, 1007, 590, 460, 84</document_evaluation_pool>
<document_relevance_vector>1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1</document_relevance_vector>
<no_relevant_documents>80</no_relevant_documents>
</пребарување>
- <пребарување topic="18">
<search_пребарување>македонија сообраќај горан николов</search_пребарување>
<desc>Сите тикети за Македонија Сообраќај по кои работел Горан Николов.</desc>
<narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет за коминтентот Македонија Сообраќај. Дополнително тикетот е релевантен само доколку лицето Горан Николов на било кој начин е вклучено во

обработката на тикетот.</narrative>
<пребарување_category>**DB**</пребарување_category>
<document_evaluation_pool>**498, 1180, 2648, 829, 305, 465, 830, 662, 306, 1011, 1041, 656, 820, 301, 822, 496, 585, 495, 930, 888, 300, 821, 721, 2898, 304, 657, 769, 254, 1434, 2641, 103, 275, 1503, 584, 145, 823, 2483, 1362, 744, 1363, 2649, 666, 1884, 665, 331, 816, 1464, 329, 322, 771, 1768, 2110, 2650, 174, 82, 590, 36, 100, 754, 1026, 618, 1788, 2158, 271, 494, 1308, 307, 52, 84, 83, 499, 1399, 37, 522, 166, 2157, 582, 1300, 583, 521**</document_evaluation_pool>
<document_relevance_vector>**1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1**</document_relevance_vector>
<no_relevant_documents>**80**</no_relevant_documents>
</пребарување>
- <пребарување topic="**19**">
<search_пребарување>**АртАИИС**</search_пребарување>
<desc>**Сите тикети за продуктот АртАИИС.**</desc>
<narrative>**Документот е релевантен (се означува со 1) доколку е од тип Тикет за продуктот АртАИИС.**</narrative>
<пребарување_category>**DB**</пребарување_category>
<document_evaluation_pool>**36, 300, 33, 301, 499, 820, 683, 326, 465, 69, 656, 697, 716, 816, 780, 174, 747, 305, 771, 584, 821, 621, 657, 512, 753, 145, 28, 630, 696, 271, 823, 663, 851, 778, 722, 67, 700, 776, 307, 834, 699, 779, 517, 718, 586, 831, 29, 460, 721, 265, 666, 717, 83, 835, 585, 719, 830, 694, 590, 781, 495, 664, 695, 37, 662, 35, 634, 66, 665, 580, 829, 4, 577, 575, 52, 772, 65, 846, 837, 82**</document_evaluation_pool>
<document_relevance_vector>**1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1**</document_relevance_vector>
<no_relevant_documents>**80**</no_relevant_documents>
</пребарување>
- <пребарување topic="**20**">
<search_пребарување>**ненад стоевски 12/2013**</search_пребарување>
<desc>**Сите тикети кои референцираат на Ненад Стоевски во месец декември 2013 година.**</desc>

<narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет, а датумот на пријава или пак разрешување мора да биде во декември 2013 година. Дополнително тикетот е релевантен само доколку истиот референцира на вработеното лице Ненад Стоевски.</narrative>
<пребарување_category>DB</пребарување_category>
<document_evaluation_pool>2653, 2731, 2705, 2640, 2698, 2700, 2617, 2708, 2672, 2628, 2638, 2784, 2658, 2730, 2732, 2736, 2751, 2764, 2753, 2711, 2709, 2616, 2671, 2627, 2694, 2657, 2717, 2614, 2662, 2691, 2608, 2710, 2635, 2636, 2735, 2703, 2695, 2704, 2679, 2655, 2747, 2682, 2651, 2706, 2722, 2724, 2629, 2752, 2712, 2620, 2746, 2666, 2610, 2734, 2701, 2643, 2733, 2654, 2632, 2707, 2613, 2633, 2644, 2634, 2716, 2680, 2745, 2618, 2639, 2661, 2681, 2702, 2773, 2719, 2645, 2663, 2692, 2652, 2738, 2667</document_evaluation_pool>
<document_relevance_vector>1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1</document_relevance_vector>
<no_relevant_documents>80</no_relevant_documents>
</пребарување>
- <пребарување topic="21">
<search_пребарување>марија тарчуговска оперативна поддршка</search_пребарување>
<desc>Сите тикети од тип оперативна подршка кои референцираат на Марија Тарчуговска.</desc>
<narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет, а типот на барањето е Оперативна поддршка. Дополнително тикетот е релевантен само доколку реферира на вработеното лице Марија Тарчуговска.</narrative>
<пребарување_category>DB</пребарување_category>
<document_evaluation_pool>1426, 1074, 946, 1564, 1394, 747, 630, 1271, 1425, 1626, 1478, 1097, 1361, 950, 719, 1276, 1625, 1269, 1477, 949, 1422, 1448, 1538, 1640, 947, 1296, 1567, 989, 1480, 948, 1119, 1569, 1092, 1293, 1546, 1064, 753, 575, 1378, 1421, 1118, 1310, 1297, 826, 1031, 1309, 1063, 1536, 1039, 1060, 1078, 1076, 700, 1447, 1423, 1058, 1623, 1461, 1420, 952, 1277, 1275, 1009, 1035, 716, 621, 1602, 1371, 1054, 1238, 951, 718, 986, 1079, 1330, 1427, 717, 1474, 2611, 1631</document_evaluation_pool>
<document_relevance_vector>1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0,

1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1</document_relevance_vector>

<no_relevant_documents>**57**</no_relevant_documents>

</пребарување>

- <пребарување topic="**22**">

<search_пребарување>**конто барање дополнување**</search_пребарување>

<desc>**Сите тикети од тип барање за дополнување кои се однесуваат на работа со конта.**</desc>

<narrative>**Документот е релевантен (се означува со 1) доколку е од тип Тикет, а типот на барањето е Барање за дополнување. Дополнително тикетот е релевантен само доколку се однесува на работа со конта.**</narrative>

<пребарување_category>**DB**</пребарување_category>

<document_evaluation_pool>**1703, 1997, 1299, 929, 1087, 404, 1398, 889, 383, 1308, 780, 2596, 252, 830, 1432, 2156, 1888, 699, 329, 68, 292, 1886, 1887, 2177, 845, 1919, 1397, 2002, 2527, 992, 864, 4, 615, 15, 1436, 1730, 65, 1198, 1011, 82, 268, 199, 1285, 696, 1393, 1762, 663, 2402, 2594, 69, 16, 36, 1441, 1395, 271, 1287, 657, 1542, 1106, 1026, 1046, 322, 1294, 1362, 83, 2153, 1430, 1141, 2664, 103, 460, 3, 695, 72, 1045, 959, 1206, 98, 1777, 2866**</document_evaluation_pool>

<document_relevance_vector>**0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0**</document_relevance_vector>

<no_relevant_documents>**5**</no_relevant_documents>

</пребарување>

- <пребарување topic="**23**">

<search_пребарување>**Драган Милисов**</search_пребарување>

<desc>**Сите тикети евидентирани од Драган Милисов.**</desc>

<narrative>**Документот е релевантен (се означува со 1) доколку е од тип Тикет, а е евидентиран од лицето Драган Милисов.**</narrative>

<пребарување_category>**DB**</пребарување_category>

<document_evaluation_pool>**950, 781, 719, 2391, 1727, 1423, 2676, 1016, 1073, 1557, 2265, 2390, 1556, 2675, 621, 953, 2112, 1714, 580, 700, 2677, 1422, 1129, 834, 889, 952, 967, 1017, 1072, 1424, 2387, 1957, 780, 1559, 2266, 2114, 717, 577, 1777, 851, 831, 776, 1544, 2421, 949, 699, 634, 837, 1959, 718, 1496, 575, 779, 1421, 835, 512, 586, 747, 1781, 1203, 2422, 716, 778,**

954, 1543, 1958, 1130, 1201, 1420, 1730, 1419, 1811, 1007, 951, 948, 852, 1571, 1964, 2420, 1726</document_evaluation_pool>
<document_relevance_vector>1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1</document_relevance_vector>
<no_relevant_documents>80</no_relevant_documents>
</пребарување>
- <пребарување topic="24">
<search_пребарување>ана паневска висок приоритет</search_пребарување>
<desc>Сите тикети од висок приоритет кои реферираат на Ана Паневска.</desc>
<narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет, а има висок приоритет. Дополнително тикетот е релевантен само доколку реферира на вработеното лице Ана Паневска.</narrative>
<пребарување_category>DB</пребарување_category>
<document_evaluation_pool>2400, 2340, 1248, 1012, 846, 65, 1542, 722, 70, 2646, 2572, 72, 2329, 326, 157, 1664, 29, 994, 944, 2342, 1216, 2155, 154, 2002, 1039, 987, 2693, 2352, 946, 772, 2623, 2853, 1573, 810, 69, 1969, 1218, 1064, 67, 187, 2699, 619, 2354, 1177, 66, 2331, 889, 1046, 1269, 2825, 2619, 68, 71, 1074, 2611, 151, 2612, 2383, 1762, 1751, 2332, 1191, 2621, 526, 28, 1097, 1063, 1380, 142, 785, 1927, 2389, 1141, 899, 2333, 2579, 967, 992, 2854</document_evaluation_pool>
<document_relevance_vector>0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0</document_relevance_vector>
<no_relevant_documents>13</no_relevant_documents>
</пребарување>
- <пребарување topic="25">
<search_пребарување>договор</search_пребарување>
<desc>Сите тикети за коминтенти кои имаат договор за одржување.</desc>
<narrative>Документот е релевантен (се означува со 1) доколку е од тип Тикет, а е пријавен од страна на коминтент кој има потпишано договор за одржување.</narrative>
<пребарување_category>TEXT</пребарување_category>
<document_evaluation_pool>2447, 1824, 946, 1800, 1601, 1657, 1074, 1738, 1271,

1277, 1191, 1344, 1039, 1580, 187, 1709, 1689, 1097, 1791, 1835, 1218, 994, 1064, 1461, 1623, 1631, 1625, 157, 1837, 1269, 1641, 1356, 1624, 1012, 1535, 1662, 1833, 1626, 1713, 1276, 1569, 1477, 1750, 1753, 1831, 1602, 526, 1659, 151, 1834, 619, 1447, 1736, 154, 1830, 1783, 142, 1797, 1378, 1836, 1769, 1063, 1704, 1537, 1729, 1819, 1330, 1361, 1749, 1360, 1828, 1560, 1370, 1826, 1426, 1710, 634, 1737, 1771, 1822</document_evaluation_pool>

<document_relevance_vector>**1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1**</document_relevance_vector>

<no_relevant_documents>**80**</no_relevant_documents>

</пребарување>

</queries>