Analys av insamling och bearbetande av audiodata från uppkopplade mobila enheter

Robert Kantero 1800501
Kandidatavhandling i datateknik
Handledare: Jerker Björkqvist
Fakulteten för naturvetenskaper och teknik
Åbo Akademi
2021

Abstract

Lorem ipsum dolor...

Innehåll

Kapitel 1

Introduktion

Internet har explosionsartat ökat mängden data som finns i världen. It-industrins jättar kan nytta väldigt mycket av att effektivt kunna analysera dessa data. I dagens läge har så gott som alla en smarttelefon eller flera med sig dygnet runt, vecka ut och vecka in, månaden i ända under hela året, alla år i framtiden. Alla har säkert befunnit sig i en situation där man fått reklam för något helt oförväntat, efter att man talat om saken med kompisar bara några timmar eller dagar tidigare. Det kan kännas som om ens telefon lyssnar på och man inte riktigt vet vad eller vem som ligger i andra ändan.

Denna avhandling kommer att försöka ge svar på frågan om teknologijättarna konstant lyssnar på oss via smarttelefoner och uppkopplade hemassistenter. Frågor som om det överhuvudtaget är tekniskt möjligt och om det vore lönsamt, ifall det de facto visade sig möjligt, kommer att vara centrala.

Strukturen på avhandlingen kommer att se ut som följande. I kapitel 2 introduceras de olika typerna av data som finns. Därefter tas de vanligaste databaserna upp i kapitel 3. Kapitel 4 berättar om ett prototypsystem för insamling av ljuddata och konvertering till textdata som möjliggör textanalys, vilket kommer att tas upp i kapitel 5. Efter att dessa aspekter tagits upp, bör det ännu i kapitel 6 talas om mängden data, energi och pengar som cirkulerar runt it-industrin. Slutligen öppnas diskussionen i kapitel 7, där den centrala frågeställningen förhoppningsvis får några tillfredsställande svar.

Kapitel 2

Data

Före man kan analysera insamlade data, är det viktigt att veta vad man har för data till förfogande. I detta kapitel redogörs de olika typerna av data som förekommer i dataveten-skap och data-analys.

2.1 Strukturerade data

Strukturerade data går hand i hand med kvantitativa data. Det är frågan om data som kan representeras med siffror och värden, data som kan användas för statistisk analys. Ett typiskt Excel kalkylark består av strukturerade data. Man kan skapa väldefinierade modeller, till vilka de olika datapunkterna från en datamängd placeras och passar in på ett sätt som går att använda. Med hjälp av väldefinierade modeller som spjälker upp större mängder data, kan man sedan gruppera om de mindre datapunkterna och forma relationer. Det är grunden för hur relationsdatabaser fungerar, mera om dem i kapitel 3. Typiska exempel på strukturerade data är namn, adresser, datum och bankkortsnumror[1].

2.2 Ostrukturerade data

Ostrukturerade data går igen hand i hand med kvalitativa data. Jämfört med strukturerade data som man kan definiera modeller för, kan man inte designa datamodeller som kunde kategoriskt placera rätt datapunkter till rätt ställe. Det är lättare att förstå varför det inte går att kategorisera ostrukturerade data lika som strukturerade data, om man betraktar några exempel.

I ett frågeformulär kan man se frågor som ska besvaras med ett numeriskt värde i en viss räckvidd. De är lätta att kategorisera och anses som strukturerade data. Frågor som är öppna för besvararen att uttrycka sig själv i ord kan däremot inte ges exakta värden. Det är inte heller någon vits med att spara videofiler i en tabellstruktur, då man inte kan

jämföra dem med varandra. Dessa sorters data kallas ostrukturerade och lämpar sig inte för användning i relationsdatabaser. Därför utvecklades NoSQL databaser som också kommer att tas upp i kapitel 3[2].

2.3 Semistrukturerade data

Som namnet tyder på, ligger semistrukturerade data någonstans mellan strukturerade och ostrukturerade data. Semistrukturerade data har något, eller några definierande eller konsekventa karaktärsdrag, utan att kunna modelleras för att passa in i ett strikt schema med tabellstruktur som i relationsdatabaser. Även om dessa data är ledigare, har de ändå vissa organisatoriska egenskaper som semantiska taggar och metadata, som hjälper med organiseringssvårigheter.

Bra exempel på semistrukturerade data är e-post. Även om innehållet i meddelandet är ostrukturerat, finns det även strukturerade data att utvinna, så som sändarens och mottagarens namn och e-post-adresser, samt tidpunkt för sändningen[2]. Mycket vanliga exempel på semistrukturerade data inom it-världen är JSON och XML[1].

2.4 Metadata

Metadata beskriver andra data genom att ge information om innehållet av posterna. Ta en bild som exempel. Metadata kunde bestå av bildens filstorlek, upplösning, datum, plats, tid, med mera. Ofta används metadata i form av metataggar i webbsidor, för att snabbt kunna ge sökmotorer en aning om vad sidan innehåller[3].

Kapitel 3

Lagring av data

3.1 Typer av databaser

På grund av de olika typerna av data som finns att behandla, behövs det olika sorts databaser för att hantera de olikt strukturerade data. I detta kapitel tas de mest använda databassystemen fram, förklarar deras struktur, samt jämför deras styrkor och svagheter sinsemellan.

3.1.1 Relationsdatabaser

Relationsdatabaser är en typ av databaser som lagrar och ger tillgång till datapunkter som är relaterade till varandra. Relationsdatabaser anses vara intuitiva och rakt på sak. Data representeras i form av tabeller av rader och kolumner. Varje rad har ett unikt ID, som kallas nyckel. Tabellens kolumner innehåller attributen av data, som för de unika raderna oftast har ett värde. Detta tillåter enkelt skapande av relationer mellan datapunkter. Relationerna kan skapas genom att köra join operationer på olika tabeller. Resultatet är en ny tabell med den specifika informationen man ville ha från de ursprungliga tabellerna.

id	name	age	sex
0	John Doe	30	M
1	Jane Doe		F

Figur 3.1: Exempel på en tabell i en relationsdatabas

Relationsdatabasernas starka sida är just användningen av tabeller, eftersom de är väldigt effektiva och flexibla på att lagra strukturerad information. En annan styrka uppkom på 1970-talet, då SQL blev utvecklat. SQL är ett språk baserat på relationsalgebra. Förkortningen kommer från engelskans Structured Query Language, alltså strukturerat fråge-

språk. SQL tillåter standardiserade frågor, vilket underlättar arbetet och ökar prestandan på databasfrågor.[4]

3.1.2 NoSQL databaser

I och med den överväldigande mängden data som blev tillgänglig i samband med uppkomsten av internet, behövde det utvecklas ett nytt sätt att hantera data som var effektivare och mer flexibelt. NoSQL, "Not only SQL" eller "inte bara strukturerat frågespråk", innebär just vad namnet säger. Utöver strukturerade data, kan man även hantera semi- eller ostrukturerade data. NoSQL modellen är icke-relationell, vilket tillåter snabb prestanda, spontana metoder för organisering, samt förmågan att bearbeta stora volymer av data. Kombinationen av att kunna hantera ostrukturerad data och dess förmåga att hantera stora volymer data snabbt, möjliggör NoSQL att bearbeta stordata effektivt. Den distribuerade arkitekturen ger stor datasäkerhet, genom säkerhetskopior av data. Flera noder (databassevrar) har överlappande data till hands, så ifall en eller flera noder genomgår driftsavbrott, fortsätter resten av noderna fungera normalt utan någon förlust av data. Med korrekt implementation kan man således erbjuda hög prestanda på stor skala med kontinuerlig drifttid.[5]

3.1.2.1 Dokumentorienterade databaser

Dokumentorienterade databaser innehåller, hanterar och hämtar semistrukturerade data.[5] Dokumenten innehåller sökbar information, inte olikt raderna i relationsdatabasernas tabeller. Skillnaden är att ordningen på informationen kan variera bland olika dokument, det kan inte finnas tomma fält utan värden och information kan lätt adderas till dokument.[6] Dokumenten kan ses som självständiga enheter, vilket ökar prestanda och gör det lättare att dela data mellan servrar. Dokumentbaserade databaser har ofta kraftiga frågemotorer som gör frågeformuleringar snabba och lätta. Detta kan ses i figur 3.3. Dokumenten i denna typ av databas är kompletta, dvs det finns inte övrig information om ett objekt som inte skulle ingå i dess dokument. Dokumenten kan således se mycket olika ut.[5] Ett exempel på dokumentorienterade databaser är MongoDB, som bl.a. Google, ebay, och Storbritanniens regering använder.[7] Figur 3.2 är ett exempel på hur man kunde spara personuppgifter i MongoDB.

```
{
  "_id": "5cf0029caff5056591b0ce7d",
  "firstname": "Jane",
  "lastname": "Wu",
  "address": {
    "street": "1 Circle Rd",
    "city": "Los Angeles",
    "state": "CA",
    "zip": "90404"
  },
  "hobbies": ["surfing", "coding"]
}
```

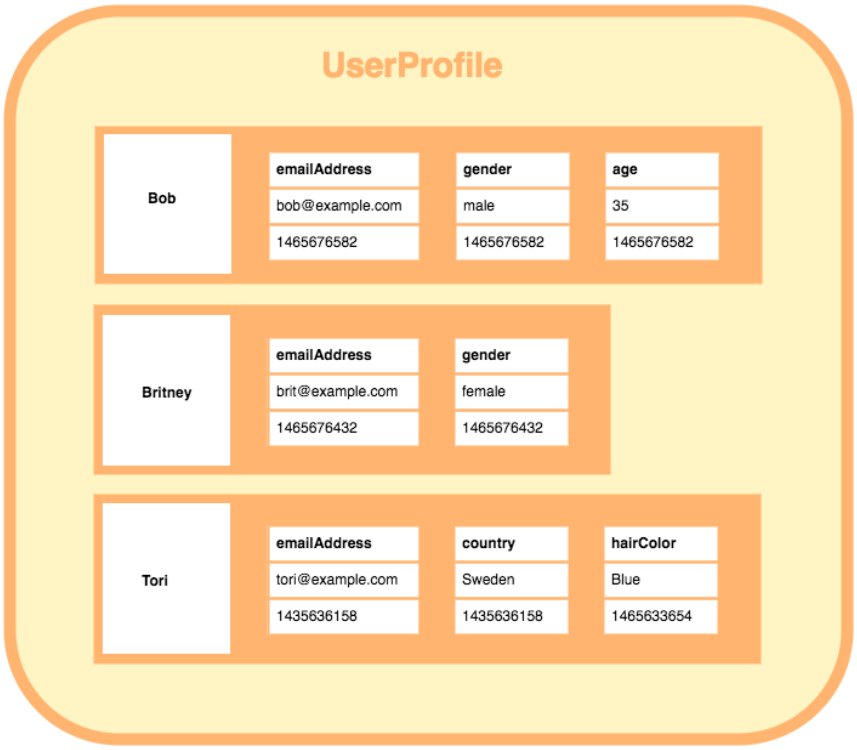
Figur 3.2: JSON-lik dokumentrsuktur i MongoDB
[8]

```
> db.users.find({ "address.zip" : "90404" })
{ "_id": "5cf0029caff5056591b0ce7d", "firstname": "Jane",
 "lastname": "Wu", "address": { "zip": "90404" } }

 { "_id": "507f1f77bcf86cd799439011", "firstname": "Jon",
  "lastname": "Davis", "address": { "zip": "90404" } }

  { "_id": "5349b4ddd2781d08c09890f3", "firstname": "Jim",
  "lastname": "White", "address": { "zip": "90404" } }

  { "_id": "5bf142459b72e12b2b1b2cd", "firstname": "Jeff",
  "lastname": "Taylor", "address": { "zip": "90404" } }

  { "_id": "5cf003283b23d04a40d5f88a", "firstname": "Jerry",
  "lastname": "Miller", "address": { "zip": "90404" } }
```

Figur 3.3: Kraftig frågemotor hittar lätt relevant information
[8]

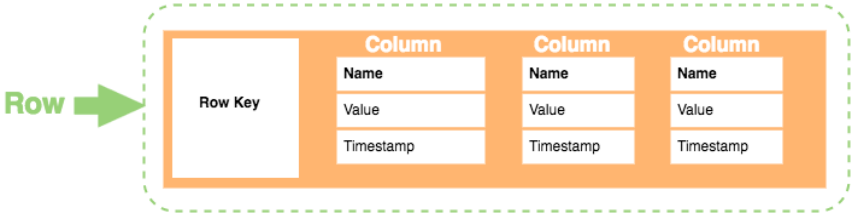
3.1.2.2 Kolumnbaserade databaser

Kolumnbaserade, eller kolumnära, databaser är relativt lika relationsdatabaser. Strukturen hos kolumnära databaser byggs upp av kolumnfamiljer. De byggs i sin tur upp av rader, som inte behöver vara likadana jämfört med de andra raderna i kolumnfamiljen. Exempel på en kolumnfamilj kan hittas i figur 3.4.



Figur 3.4: Exempelstruktur på en kolumnfamilj
[9]

Raderna byggs vidare upp av kolumner, där skillnaderna alltså kan komma fram, t.ex. i form av antal kolumner eller datatyp. Dessa kolumner är begränsade till sin egen rad, till skillnad från relationsdatabaser. Varje rad har en egen unik nyckel, som används för identifiering. Raden har ett givet antal kolumner i sig, där varje kolumn består av ett namn/värde-par och en tidsstämpel för att identifiera den nyaste uppdateringen. Ett recept för en generisk rad i en kolumnfamilj kan ses i figur 3.5.

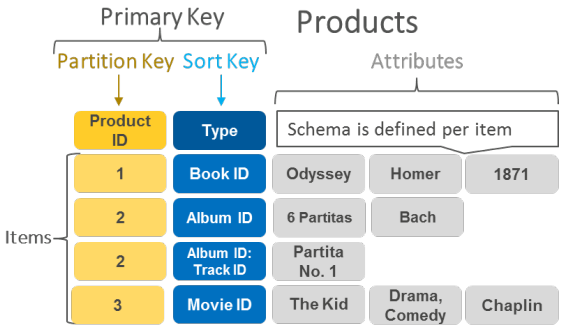


Figur 3.5: Exempelrad i en kolumnfamilj
[9]

Vissa kolumnära databaser tillåter även nästlande av objekt inuti kolumnerna. Tack vare sin struktur är dessa databaser väldigt effektiva på datakomprimering, partitionering och aggregationsfrågor.[9]

3.1.2.3 Nyckel-värde-databaser

Nyckel-värde-databaser är byggda enligt en väldigt enkel princip, där ett värde associeras med en unik nyckel. Värdet kan i allmänhet vara vad som helst, från en enkel sträng eller ett tal, till räckor eller t.om. objekt. Man kan jämföra ett nyckel-värde par till en räcka eller en map från diverse programmeringsspråk, där man t.ex. kan definiera en räcka “x” att bestå av heltalen från 0-10. I detta fall skulle “x” vara nyckeln och värdet är “0,1,2,3,4,5,6,7,8,9,10”. Eftersom principen på denna modell av databas är så simpel, är det ofta ändamålsenligt att också spara relativt simpla värden åt de effektivt indexerade nycklarna, för att optimera högt dataflöde.



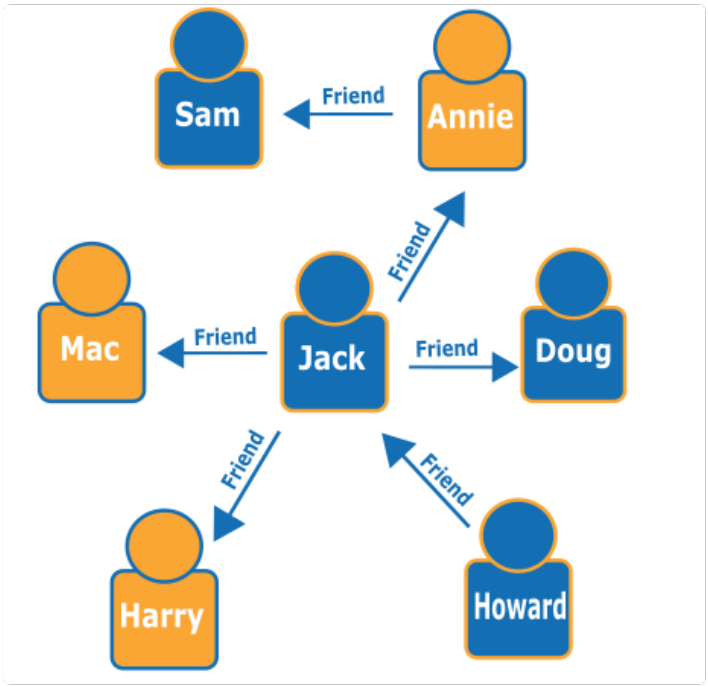
Figur 3.6: Exempel på DynamoDBs nyckel-värdestruktur
[10]

Nyckel-värde-databasers namn definierar dem, nämligen deras grundläggande funktion är att ge användare eller program tillgång till data med hjälp av en nyckel. Eftersom definitionen är så enkel, men det är lätt att utöka och optimera funktionalitet, är det omöjligt att ha en universal lista av egenskaper hos dessa databaser. De mest elementära funktionerna som i regel kan tas för givet är att hämta, sätta, uppdatera, ersätta, eller radera

värden associerade med en nyckel.[11]

3.1.2.4 Grafdatabaser

Grafer är byggda av noder och bågar, där bågarna kopplar noderna samman enligt något förhållande. Samma princip utnyttjas i grafdatabaser för att hålla förhållandena lika viktiga som själva datapunkterna. Noderna representeras här av entiteter och bågarna representeras av entiteternas förhållande till varandra. En entitet är något som existerar oberoende av sina attribut. Som ett vardagligt exempel betyder det att oberoende om en person byter sitt namn, är hon ändå samma person. Varje båge måste ha en startnod och en slutnod, såväl som en typ och en riktning. Bågen kan representera vilken sorts förhållande som helst. Ett lätt exempel, som utnyttjas till hög effekt bl.a. i rekommendationsmotorer, är vänskap. Detta kan ses i figur 3.7. Det finns ingen gräns på hur många förhållanden en entitet kan ha, eller hurdana dessa förhållanden kan vara.[12] Grafdatabaser presterar bättre än övriga databaser på att bevara, behandla och fråga om förhållanden. Eftersom grafdatabaser sparar förhållandena tillsammans med datapunkterna, behöver den inte beräkna samkörningar, joins, vid varje fråga. Detta ökar effektiviteten och tillåter miljontals av noder nås per sekund per kärna, utan att behöva ta irrelevant data utanför frågan i beaktande.[13]



Figur 3.7: Howard är vän med Jack, som är vän med Annie, som är vän med Sam. Det är lätt för en rekommendationsmotor att rekommendera Sam som vän åt Howard.
[12]

3.2 Jämförelse mellan olika databaser

Efter mycket tekniska förklaringar om hur dessa olika typer av databaser fungerar, kan det vara till skäl att snabbt upprepa deras olika för- och nackdelar.

3.2.1 Relationsdatabaser

Relationsdatabaser fungerar bra för strukturerade och förutsägbara data, då man kan behandla dessa data på flera olika sätt. Databasens schema bestämmer hur tabellerna byggs upp, vilket gör det krångligt att ändra strukturen i databasen. Samtidigt är det en bra sak, nämligen det upprätthåller dataintegriteten.[14] Relationsdatabaser stöder ACID-principen (från engelskans Atomic, Consistent, Isolated, Durable)[15][16], dvs. att varje transaktion är atomär, konsistent, isolerad och hållbar. I praktiken betyder detta att en transaktion antingen görs i sin helhet eller inte alls, att databasen befinner sig i ett konsistent tillstånd efter en transaktion, att en transaktion utförs som om den vore den enda transaktionen som utförs i databasen och att då en transaktion är slutförd går den inte att ta tillbaka[15]. Skalbarheten lider dock vågrätt i relationsdatabaser[16]. Vågrät skalbarhet innebär ökande av resurser i form av fler maskiner, medan lodrät skalbarhet innebär ökning av resurser genom att förse den existerande maskinen med mer eller bättre hårdvara[17].

Fördelar

- Fungerar väl med strukturerade data
- Inbyggd dataintegritet
- Stöder ACID
- SQL
- Oändlig indexering

Nackdelar

- Skalar inte vågrätt
- Normaliserad data leder till flera joins, drabbar hastighet
- Problem med semistrukturerade data

[16]

3.2.2 NoSQL databaser

NoSQL databaser kan variera från varandra i hög grad. De absolut flesta har dock ett gemensamt problem med att hålla databasen konsistent. Enligt professor Eric Brewers CAP-teorem kan inte webbtjänster samtidigt förse konsistens, tillgänglighet och partitionstolerans. NoSQL databaser löser problemet med att lösa på konsistensens restriktioner, med hjälp av BASE (från engelskans Basically Available, Soft state, Eventually consistent). BASE lägger högre prioritet på tillgänglighet, medan konsistensen blir svagare. Utan ändringar i databasen skulle den eventuellt bli konsistent, men det är inte målet, då man vill ha en databas som är tillgänglig hela tiden[15].

Fördelar

- Hög vågrät skalbarhet
- Fungerar väl med semi- och ostrukturerade data
- Högre tillgänglighet (BASE)
- Vissa kan erbjuda ACID stöd

Nackdelar

- Lägre, eller eventuell konsistens (BASE) jämfört med ACID
- Begränsat stöd för joins
- Ingen inbyggd dataintegritet, den måste kodas
- Ändlig indexering

[16]

3.2.2.1 Dokumentorienterade databaser

Skillnaden mellan högt avancerade nyckel-värde-databaser och dokumentorienterade databaser kan vara svår att urskilja ibland, då båda tillåter en mängd data associeras med en nyckel. Dokumentorienterade databaser har dock den värdefulla skillnaden att data sparas i en dokumentstruktur, vilket tillåter frågor och analys av sparade data. Eftersom dokumenten är självständiga och kompletta, kan man lätt ändra på dokumenten utan oro för att ändra på databasens struktur eller andra data.

Användning av dokumentorienterade databaser kan löna sig då man har en mängd diskreta data, som inte behöver korsrefereras mycket. Den höga flexibiliteten på databasens användning innebär dock att man själv är ansvarig för strukturen och konsistensen för databasen[14].

3.2.2.2 Kolumnbaserade databaser

Kolumnbaserade databaser är på ytan väldigt lika relationsdatabaser, men man ser snabbt skillnaden då man jämför databasernas struktur. Genom att jämföra figur 3.1 med figur 3.4, kan man se hur flexibla kolumnfamiljerna kan vara.

Man kan också dra liknelser till en nyckel-värde-databas, eftersom kolumnbaserade databaser är så radinriktade. Raden har en nyckel som kan innehålla helt andra data än raden ovanför.

Kolumnbaserade databaser är ett bra verktyg då man behöver hög skalbarhet och prestanda på en radnivå. Eftersom en rad innehåller all information om ett objekt, både data och metadata, behövs inga joins för att hämta relevant information. Radstrukturen förvaras också typiskt på samma maskin i ett kluster, vilket gör skärvning och skalande mindre komplicerat. Däremot är det just den höga inriktningen på radbaserade operationer som gör att kolumnbaserade databaser presterar dåligt, ifall man har väldigt sambandstunga data som kräver mycket joins. Analytiska operationer som att summera flera olika posters data är ofta svåra, om inte rakt ut omöjliga.[14]

3.2.2.3 Nyckel-värde-databaser

På grund av sin simpla struktur, är nyckel-värde-databaser oftast använda för simpla ändamål. Deras styrka är snabbhet, inte bara för att man använder dem i oinvecklade sammanhang, men också för att hela datamängden laddas direkt in i minnet, i stället för att vara tvungen att hämta från en lagringsenhet per operation[14].

3.2.2.4 Grafdatabaser

Då man talar om relationsdatabaser, är det viktigt att inse att ordet relation refererar till förmågan att förknippa information från olika tabeller. Då man talar om grafdatabaser, är själva idén att skapa och behandla förhållanden mellan poster. Ofta används ordet förhållande i stället för relation, för att undvika missförstånd.

Grafdatabaser är best utnyttjade i datamängder där förhållanden och samband är av hög vikt. I stället för att vara tvungen att köra joins på tabeller i en relationsdatabas för att hitta någon gemensam datapunkt, är det trivialt i en grafdatabas, som själv avbildar dessa samband[14].

Kapitel 4

Livslogg

I följande kapitel kommer ett prototypsystem för att samla in ljud från smarttelefoner och bilda en komplett livslogg med hjälp av ljud- och textfiler att tas upp.

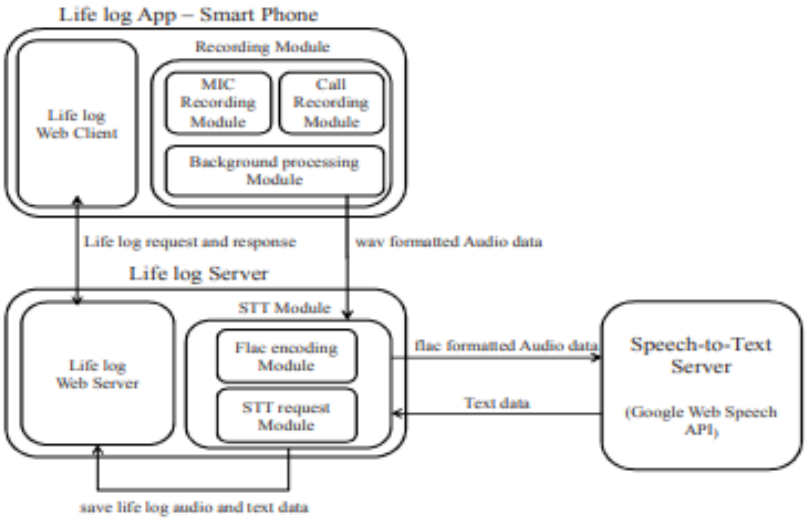
4.1 Koncept

Ett prototypsystem för att samla in ljud från smarttelefoner och konvertera dessa till textfiler implementerades år 2014. Konceptet för systemet är att använda sig av smarttelefnonens inbyggda mikrofoner för att banda in användarens röst såväl allmänt som också i telefonsamtal. De bandade ljudfilerna skickas till en server som konverterar ljudfilerna och begär text från en tal-till-textserver (TTT-server). En loggserver parar ihop ljudfilerna med sina respektive dikterade textfiler från TTT-servern. Data från loggservern erbjuds via en webbservice för webbläsare och smarttelefoner. Användarna av systemet kan således söka och utnyttja data ur sina liv.

4.2 Arkitektur

Mjukvaruarkitekturen på det ovannämnda livsloggsystemet fungerar enligt följande. Systemet består av tre huvudsakliga delar, en livslogg app för smarttelefonen, en livsloggserver och en tal-till-textserver. Appen på smarttelefonen fungerar både som klientprogram och datainsamlare. I allmänna situationer bandar en MIC-inspelningsmodul ljud med hjälp av smarttelefonens mikrofoner, medan vid telefonsamtal en skild telefonsamtalsinspelningsmodul bandar ljudet. Dessa två moduler är kontrollerade av en bakgrundsprocessormodul, som är i kontakt med livsloggservern. Inspelningsmodulerna generar en minut långa .WAV-filer, som processormodulen skickar till livsloggservern och sedan raderar från smarttelefonen.

Livsloggservern hanterar livsloggdata, ljudfil-textfil-par, samt erbjuder en webbservice för användargränssnitt. För att generera motsvarande textfiler för ljudfilerna appen skickat, används TTT-modulen. Först konverterar en Flac kodomvandlarmodul de inkommande .WAV-filerna till .FLAC-filer, varefter de skickas till en TTT-server hos Google, som returnerar textdata åt TTT-modulen. Nu kan TTT-modulen para ihop rätt ljud- och textfiler med varandra och erbjuda livsloggdata åt klientprogrammet via livsloggservern.



Figur 4.1: Koncept på mjukvaruarkitekturen för livslogg systemet
[18]

4.3 Loggade data

Eftersom ljud bandas in hela tiden av någondera inspelningsmodulen, är det kritiskt att hitta en balans mellan samplingsfrekvens, buffertstorlek och diktationsnoggrannhet, för att inte överbelasta smarttelefonernas begränsade minnen, mobila uppkopplingshastigheter, eller livsloggserverns kapacitet. Det visar sig att en god balans nås vid 16kHz samplinggsfrekvens och 256KB buffertstorlek, vilket leder till ca 68% diktationsnoggrannhet. Då man kombinerar filstorlekarna med två olika filtreringsmetoder för att rensa bort onödiga filer, når man slutligen konkreta siffror på hur mycket data man blir tvungen att lagra. En möjlighet är att sätta en tröskel på 45 decibel och radera filpar vars ljudfil inte når tröskeln. Den andra metoden är att radera filpar vars textfiler är 0 byte stora. Med de ovannämnda parametrarna loggas enligt experimentella resultat dryga 43GB och 12GB för ljudnivåfiltret respektive textfilfiltret under en månad per användare[18].

Kapitel 5

Textutvinning

Textutvinning är svårt på grund av flera orsaker, men nya metoder utvecklas för att göra processen mer effektiv[19].

Kapitel 6

Datacenter

6.1 Elförbrukning

6.1.1 Förbrukning per datacenter

För att driva ett datacenter krävs det enorma mängder elektricitet. Anläggningen kan vara upp till flera hektar stor[20] och innehålla enorma mängder servrar, nätverksutrustning, kylningssystem och övrig infrastruktur. Var anläggningen finns spelar också roll, nämligen i kyligare klimat behöver man kanske inte spendera lika mycket resurser på kylning som i ett varmare klimat, eller så kan priset på elektricitet variera från ett ställe till annat.

För att minska elkonsumtionen i sina datacenter så mycket som möjligt har Google tagit till flera olika åtgärder. Högpresterande servrar skräddarsydda för att använda så lite elektricitet som möjligt används för att optimera datamängdernas orsakade elkonsumtion. För själva anläggningens konsumtion installeras smarta ljus- och temperaturkontrollsystem, såväl som optimal kraftdistribution för minimal energiförlust. Från 2015 till 2020 har beräkningsraften per använd elektrisk kraft ökat ungefär sjufaldigt[21].

6.1.2 Förbrukning globalt

Datacenter mäter sin effektivitet med hjälp av PUE (power usage effectiveness), där datacentrets totala elkonsumtion delas med it-utrustningens elkonsumtion. En PUE av 1.0 skulle ju vara ideal, då all energi går åt till servrernas verksamhet. Detta är ju inte möjligt i praktiken, men ju närmare 1.0 man kommer, desto mer effektiv anläggning har man.

På grund av ovannämnda insatser för att prestera bättre, har Googles datacenter nått en ny nivå med en globalt kollektiv PUE av 1.10, jämfört med medelvärdet inom industrin, 1.67. Google använde märkbart över 12TWh år 2019, varav den största delen användes i datacenter. Man kan grovt uppskatta att 12TWh då användes i datacenter, vilket vid en PUE av 1.10 ger 10.9TWh effektivt bruk[21].

6.2 Kostnader

Energikonsumtion per dataenhet är något som är svårt att mäta exakt, då nätverkstrafiken kan begränsas väldigt lokalt eller ruttas runt världen. Därför varierar ofta siffrorna beroende på vem som rapporterar om saken och varför. Från akademiska publikationer har allting från 0.04kWh/GB till 136kWh/GB rapporterats stämma. Äldre data går inte heller att applicera som det är till nutiden, då teknologiska framsteg har ökat prestandan, det vill säga man kan åstadkomma lika mycket med mindre elkonsumtion[22].

Enligt en studie om ändpunkt till ändpunkt it-energikonsumtion från 2012, kostar en gigabyte data på internet ca 5.12kWh, vilket motsvarar $0.51 USD. Per gigabyte trafik på internet bär datacenter 48% av belastningen, vilket motsvarar ca 2.47kWh/GB, alltså 0.25$/GB[23]. Ett mycket grovt estimat för hur mycket data Googles datacenter hanterar årligen kan då uträknas enligt

$$\frac{10.9TWh}{2.47\frac{kWh}{GB}} = 4.4EB \quad | \quad \text{Exabyte} = 1000^6 Byte.$$

På grund av det relativt konservativa måttet på energi per dataenhet, kan den totala datamängden verka lite låg. T.ex. Heshmore uppskattar år 2017 Googles datamängd att befinna sig i räckvidden 10-15EB[24]. Det betyder att deras kostnader i elkonsumtion för den behandlade datamängden kan tänkas befinna sig någonstans mellan

$$4.4 \times 10^9 GB \times 0.25\frac{\$}{GB} = \$17.7 \text{ miljarder dollar}$$

och

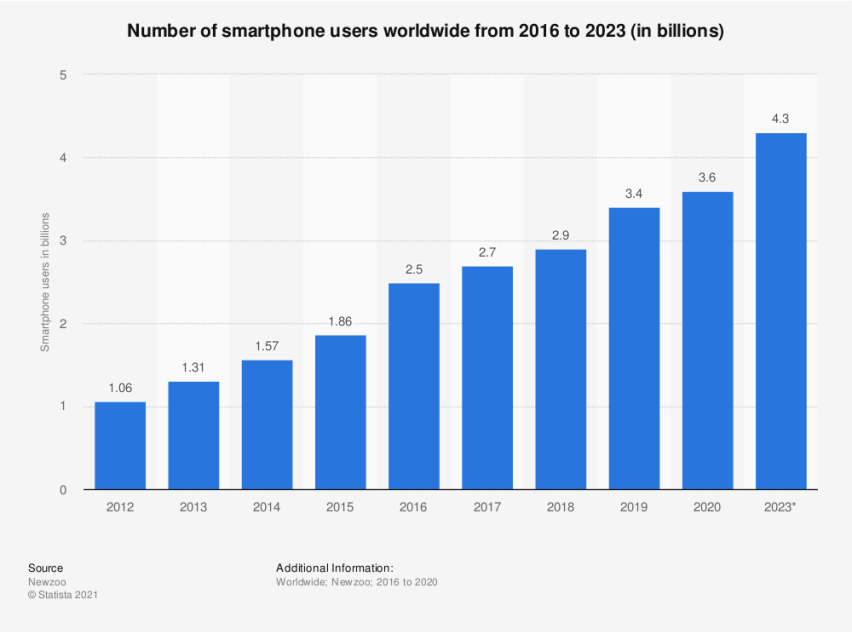
$$15 \times 10^9 GB \times 0.25\frac{\$}{GB} = \$37.5 \text{ miljarder dollar.}$$

Dock måste man också ta den amerikanska dollarns köpkraft och energipriserna i beaktande. Som konstaterat, kan man inte applicera äldre data med exakt noggrannhet. Givet inexakta data, både gällande ekonomin och datamängden, kan man inte uppskatta årliga datakostnaderna noggrannare än $15-40 miljarder dollar.

Kapitel 7

Diskussion

Här diskuteras de olika ämnena som tagits upp igenom avhandlingen tillsammans för att klargöra ifall det är möjligt att lyssna på alla mobila enheter hela tiden och ifall möjligt, om det vore lönsamt att göra det.



Figur 7.1: Antalet användare av smarttelefoner i miljarder
[25]

Från figur 7.1 ser man antalet smarttelefoner i användning. Kombinerat med data från den prototypiska livsloggens implementation kan man estimera hur mycket data som skulle genereras av konstant övervakning. Ett grovt estimat för år 2020 kan beräknas enligt

$$12\frac{GB}{\text{mån}} \times 12 \times \text{mån} \times 3.6 \times 10^9 = 520EB.$$

Uppenbarligen ka

Källförteckning

[1] Devin Pickell. *Structured vs Unstructured Data – What's the Difference?* URL: https://learn.g2.com/structured-vs-unstructured-data. (accessed 25.3.2021).

[2] Bernard Marr. *What's The Difference Between Structured, Semi-Structured And Unstructured Data?* URL: https://www.forbes.com/sites/bernardmarr/2019/10/18/whats-the-difference-between-structured-semi-structured-and-unstructured-data/. (accessed 25.3.2021).

[3] TechTerms. *Metadata*. URL: https://techterms.com/definition/metadata. (accessed 25.3.2021).

[4] Oracle. *What is a relational database?* URL: https://www.oracle.com/database/what-is-a-relational-database/. (accessed: 25.2.2021).

[5] Keith D. Foote. *A Brief History of Database Management*. URL: https://www.dataversity.net/brief-history-database-management/#. (accessed: 25.2.2021).

[6] ComputerSweden. *dokumentorienterad*. URL: https://it-ord.idg.se/ord/dokumentorienterad/. (accessed: 25.2.2021).

[7] MongoDB. *Our Customers | MongoDB*. URL: https://www.mongodb.com/who-uses-mongodb. (accessed: 23.3.2021).

[8] MongoDB. *The most popular database for modern apps | MongoDB*. URL: https://www.mongodb.com/. (accessed: 25.2.2021).

[9] Ian. *What is a Column Store Database?* URL: https://database.guide/what-is-a-column-store-database/. (accessed: 25.2.2021).

[10] Amazon Web Services. *What Is a Key-Value Database?* URL: https://aws.amazon.com/nosql/key-value/. (accessed 23.3.2021).

[11] MongoDB. *What is a Key-Value Database? | MongoDB*. URL: https://www.mongodb.com/key-value-database. (accessed: 23.3.2021).

[12] Amazon Web Services. *What Is a Graph Database?* URL: https://aws.amazon.com/nosql/graph/. (accessed 23.3.2021).

[13] Neo4j. *What Is a Graph Database? - Developer Guides*. URL: https://neo4j.com/developer/graph-database/. (accessed 23.3.2021).

[14] Prisma. *Compare database types | Database types evolve to meet different needs*. URL: https://www.prisma.io/dataguide/intro/comparing-database-types. (accessed 24.3.2021).

[15] Dan Pritchett. *BASE: An Acid alternative*. URL: https://queue.acm.org/detail.cfm?id=1394128. (accessed 24.3.2021).

[16] Keith D. Foote. *A Review of Different Database Types: Relational versus Non-Relational*. URL: https://www.dataversity.net/review-pros-cons-different-databases-relational-versus-non-relational/. (accessed 24.3.2021).

[17] Nati Shalom. *Difference between scaling horizontally and vertically for databases [closed]*. URL: https://stackoverflow.com/questions/11707879/difference-between-scaling-horizontally-and-vertically-for-databases. (accessed 24.3.2021).

[18] D. Seo, S. Kim, G. Song m. fl. "Speech-to-text-based life log system for smartphones". I: *2014 IEEE International Conference on Consumer Electronics (ICCE)*. 2014, s. 343–344. DOI: 10.1109/ICCE.2014.6776033.

[19] N. Zhong, Y. Li och S. Wu. "Effective Pattern Discovery for Text Mining". I: *IEEE Transactions on Knowledge and Data Engineering* 24.1 (2012), s. 30–44. DOI: 10.1109/TKDE.2010.211.

[20] Peggy Smedley. *Why Size Matters for Data Centers!* URL: https://connectedworld.com/why-size-matters-for-data-centers/. (accessed 30.3.2021).

[21] Google. *Google environmental report 2020*. URL: https://www.gstatic.com/gumdrop/sustainability/google-2020-environmental-report.pdf. (accessed 29.3.2021).

[22] Tom Greenwood. *Why do estimates for internet energy consumption vary so drastically?* URL: https://www.wholegraindigital.com/blog/website-energy-consumption/. (accessed 30.3.2021).

[23] D. Costenaro och A. Duer. *The Megawatts behind Your Megabytes: Going from Data-Center to Desktop*. URL: https://www.aceee.org/files/proceedings/2012/data/papers/0193-000409.pdf. (accessed 29.3.2021).

[24] maheshmore1. *How much data does google handle??* URL: https://www.heshmore.com/how-much-data-does-google-handle/. (accessed 30.3.2021).

[25] Statista. *Number of smartphone users worldwide from 2016 to 2023(in billions).* URL: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. (accessed 30.3.2021).