LØSNINGSSKISSE TIL EKSAMENSOPPGAVE I FAG TDT4305 – MAI 2018

NB! Dette er ikkje fullstendige løysingar på oppgåvene, kun skisse med viktige element, hovudsakleg laga for at vi skal ha oversikt over arbeidsmengda på eksamen, og som huskeliste under sensurering. Det er også viktig å være klar over at det også kan vere andre svar ein dei som er gjeve i skissa som vert rekna som korrekt om ein har god grunngjeving.

Oppgåve 1 – Hadoop – 10 %

a) Replikering av blokker og bruk av sjekksum på kvar blokk for å detektere diskfeil.
b) Combiner utfører delvis samanslåing av par med same nøkkel fra mapper (oftast vha. same funksjon som Reduce) , slike at desse vil verte prosessert som gruppe av Reduce-task.
   Kan redusere kommunikasjonskostnad.
c) Reduserer kommunikasjonskostnad for skrive-klient som kun sender data til ei anna node i staden for til tre, og i tillegg der ein har fleire rack vil ein redusere kommunikasjon mellom rack (om to replika vert skrive til remote rack vil ein kun ha behov for ein inter-rack kommunikasjon).

Oppgåve 2 –Spark – 10 %

Kan også besvares med Python eller Java. "Utskrift" i oppgåva er gjeve for å vise innhald i resultat-RDD, ikkje noko krav om formatering i studentane sin kode.
Generelt: Ikke trekk for mindre "språk-typo" (hensikta med oppgåva er å vise at ein forstår Spark, men trekk om det er gjeve fleire svaralternativ der eitt eller fleire er feil (forsøk på "gardering").
OK om "add" er brukt i Python i staden for "_+_", ein variant av countByKey kan også brukast istaden for reduceByKey.
Trekk for dei som ikkje forstår at det er arrays og ikkje tuplar i start-RDD.
Gjeve: val s = sc.textFile("streamed.csv").map(_.split(","))

a) val data1 = s.map(a => (a(2),1)).reduceByKey(_ + _)
b) val data2 = s.map(a => (a(1),1)).reduceByKey(_ + _)
c) val data3 = s.map(a => (a(3))).distinct().count // OK om også a(2) er med slik at artist+song er "nøkkel"

Oppgåve 3 – NoSQL – 15 %

a) Key-value stores: Data-modell basert på aksess til *verdi* (typisk post, objekt, eller dokument, men kan også ha meir kompleks data-struktur) via *nøkkel.*
   Document stores: Lagrar data som dokument i format som, t.d., JSON. Aksesserast via dokument-identifikator. Indeksering.
   Extensible record stores/BigTable clones: Tabell partisjonert i kolonne-familiar, der kvar kolonne-familie lagra i eigen fil.
   Graph Databases: Data representert som graf, der relaterte noder kan finnast ved å traversere kantar vha. sti-uttrykk.
b) Jfr. 24.4.2 i Elmasri&Navathe. Sentrale element: *sharding* og *horisontal skalerbarheit vha. konsistent hashing* (som gjev lastbalansering og feiltoleranse). Forventa at "ringen" er teikna/forklart.

Oppgåve 4 – MinHashing – 10 %

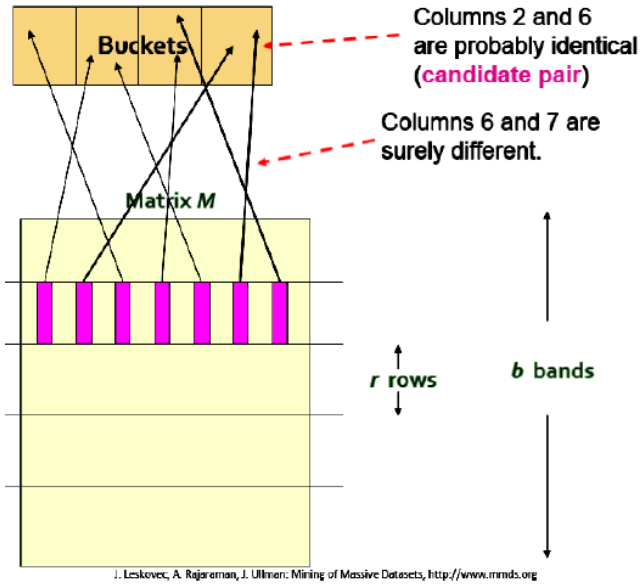
(Som det også vart informert om på eksamen så er tittelen litt misvisande, det er ikkje nødvendig å forklare shingling og MinHashing, vi er kun ute etter detaljane for LSH, og det er kun svar relatert til LSH som gjev poeng).

Jfr. 3.4 i MMD.

Hensikt: Effektivt finne dokument-par som er potensielle "nær-duplikat", alternativet er f.eks. å sjekke alle par av dokument (evt. signaturane deira) mot kvarandre.

Input: Signatur-matrise M (Docs x Signatures), Output: Liste med kandidat-par S1, S2 som må evaluerast med s(S1, S2)

Datastruktur: Matrisa M er delt inn i *b* "bands", kvar med *r* rader:



For kvart dokument, for kvar band: hash denne delen av signaturen inn i hash-bøtte (Separate bøtter for kvart band).

Gå gjennom alle bøtter for alle band: Konstruer kandidat-par for alle dokument i bøtte.

Oppgåve 5 – Adwords – 5 %

Jfr. kap. 8 i MMD:

Eit sett med bod for kvart søkeord, budsjett for kvar annonsør, eit sett med brukarar og deira søkeord.

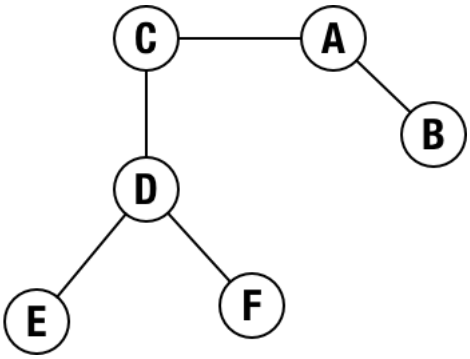
Adwords-problemet er å gje match mellom spørjing og søkeord på ein slik møte at inntekt vert maksimert.

Grådig algoritme: vel tilfeldig annonsør som har bod for søkeord i spørjing. Ulempa med denne er at ein annonsør med mange søkeord kan bruke opp budsjettet sitt og det kun er annonsørar med avgrensa (og ikkje søkte etter) søkeord att.

Endring: Vel annonsør som har høgst attverande budsjett.

Oppgave 6 (15%)

1. Beskriv og forklar kort hvilke steg du trenger å kjøre for å finne «community» i en graf. Bruk følgende figur til å støtte forklaringen din. (6%)



Svar:

In this task, it is expected that the students not only describe the algorithm but also is able to tell which method he/she uses to solve it. There are several methods for community detection. One possible method is Girwan-Newman algorithm. To get a full score the answer need to show in full that he/she understand the method.

1. Forklar hovedhensiktene med å analyser sosiale grafer generelt. (4%)

Svar:

Here the expected answer would be to show that the student understand what it means to analyse social graphs. On goal would be to identify community, e.g. to understand how users in a social network are connected. This could be useful for group recommendation, social interactions and preferences, etc. Another goal could be to simply partition graphs to reduce the amount of information to be processed (e.g. through graph clustering etc.).

Forklar hovedforskjellene mellom Storm og Spark. Forklar hvilke fordeler AsterixDB sitt Feed-system har i forhold til både Storm og Spark. (5%)

Svar:

Storm vs. Spark:

	Storm	**Spark**
Processing Model	Event-Streaming	Micro-Batching / Batch (Spark Core)
Delivery Guarantees	At most once / At least once	Exactly Once
Latency	Sub-second	Seconds

Language Options	Java, Clojure, Scala, Python, Ruby	Java, Scala, Python
Development	Use other tool for batch	Batching and streaming are very similar

AsterixDB vs. both:
- Scalable, Fault-Tolerant, and Elastic data ingestion facility
- Plug-and-Play model to cater to wide variety of data sources, through both UDF and Adapters.
- While Storm & Spark are both with custom built solution, e.g, Storm + MongoDB, AsterixDB can reduce the number of moving parts needed to handle Big Data.

Oppgave 7 (20%)

1.

a. Hva er «Bloom Filter» og hva brukes det til? Bruk eksempel til å støtte forklaringen din. (4%)

Svar:

With **a filter,** we want to accept those tuples in the stream that meet a criterion. Accepted tuples are passed to another process as a stream, while other tuples are dropped. A bloom filter is an example of such a filter. A Bloom filter consists of:
1. An array of n bits, initially all 0's.
2. A collection of hash functions h1 , h2 , . . . , hk . Each hash function maps "key" values to n buckets, corresponding to the n bits of the bit-array.
3. A set S of m key values.

The purpose of the Bloom filter is to allow through all stream elements whose keys are in S, while rejecting most of the stream elements whose keys are not in S.
For the example, refer to the lecture notes (slides).

b. Ved å bruke bloom filter-prinsippet fyll ut følgende tabell. Anta at vi skal bruke *h* som hash-funksjon, og at den er definert som følgende *h(x)= y mod 11*, der *y* er hentet henholdsvis fra oddetalls-bits fra *x* eller partalls-bits fra *x*. Eksempel*: $h_1(39) = 011 = 3$, $h_2(39) = 101 = 5$*, osv. (6%)

Strømelement	**Hash-funksjon - h_1**	**Hash-funksjon - h_2**	**Filtrere Innhold** 000 0000 0000
85 = **101 0101** 214 = **1111 1010**	1111 =15 mod 11 = 4 1100=12 mod 11 = 1	000 = 0 1111=4	**1**00 0**1**00 0000 **11** 0100 0000

353 = **01 0110 0011**	11001=25 mod 11 = 3	00101=5	**110 111**0 0001

2. Anta at du skal finne andeler av unike spørringer den siste måneden i et søkesystem fra en strøm av spørringer (search queries). På grunn av plass- og tidsbegrensninger kan du ikke sjekke alle innkomne spørringer og må derfor bruke sampling. Du bestemmer deg for å lese hver 10. spørring (dvs. 10% sampling). Forklar hvorfor dette ikke vil fungerer, dvs. hvorfor vil ikke dette gi korrekt svar? Hva ville være en mer korrekt måte å sample strømmen av spørringene på? (5%)
   Svar:
   A correct answer should at least contain the fact that sampling on every 10th query would mainly focus on the position in stream not the value of the stream element and therefore wouldn't be representative for the set of queries, i.e., the fraction of unique queries in the sample is not equal to the fraction for the stream as a whole. A solution should be focusing on the content, which can be achieved by using hashing on the value, i.e., hash search queries to 10 buckets 0, 1, … , 9. In this way, the sample is equal to all search queries that hash to bucket 0.
   All or none of the instances of a query are selected. Therefore, the fraction of unique queries in the sample is the same as for the stream as a whole.

3. Tenk at du skal bruke sosiale media som Twitter for å analysere trender o.l. Du er spesielt interessert i å vite når et produkt nevnes av folk og hvor ofte de nevnes. Du foreslår å bruke «bucket»-prinsippet til å telle hvor mange ganger dette produktet blir nevnt. Forklar hvordan «bucket»-prinsippet på datastrøm fungerer i dette tilfelle. (5%)
   Svar:
   It is important that the student shows that he/she understand the principle with bit counting. Here to be able to use bucket we have first to convert the problem to bit counting, i.e., for every tweet mentioning a specific product, we set the bit to be 1, 0 otherwise. The bucket is then a segment of the window, represented by a record consisting of:
   - The timestamp of its end [O(log2 N) bits].
   - The number of 1's between its beginning and end.
   - Number of 1's = size of the bucket.

   The constraint on the bucket sizes is that the number of 1's must be a power of 2. Thus, only O(log2 log2 N) bits are required for this count.

Oppgave 8 (15%)

2. Såkalt «*cold start*» er en utfordring når man bruker «collaborative filtering» i et anbefalingssystem. Forklar hva som menes med «cold start» og når dette kan oppstå. Drøft videre hvilke mulige løsninger som finnes for cold start-problemet. (6%)
   Svar:
   Cold-start problem is a problem that occurs when a recommender system is not able to provide recommendation because of a new item, new user, or new community as

explained in the following:
New item problem: Small number of users that rated an item, accurate prediction for this item cannot be generated.
New user problem: Small number of items rated by a user, it is unlikely that there could be an overlap of items rated by this user and active users. User- to-user similarity cannot be reliably computed.
New community problem: Without sufficient ratings, it's hard to differentiate value by personalized CF recommendations.

Possible solutions:
As the solution for new user problem:
– Displaying non-personalized recommendation until the user has rated enough
– Asking the user to describe their taste in aggregate
– Asking the user for demographic information and using ratings of other users with similar demographics as recommendations
As the solution for new item problem:
– Recommending items through non-CF techniques content
analysis or metadata
– Randomly selecting items with few or no ratings and asking user to rate those items.
As the solution for new community problem:
– Provide ratings incentives to a small "bootstrap" subset of the community, before inviting the entire community.

3. Anta følgende bruker-rating tabell.

USERS

PRODUCTS	1	2	3	4	5	6	7	8
1	1		3			5		
2			5	4			4	
3	2	4		1	2		3	
4		2	4		5			4
5			4	3	4	2		
6	1	P	3		3			2

Anta videre at du skal bruke «item-item collaborative filtering».
a. Forklar hva er forskjellen(e) mellom «item-item collaborative filtering» og «user-item collaborative filtering». (4%)

Svar:
In this subproblem I specifically invited the candidate to show how they have understood the concepts introduced in the course and reason the answer to the question based on the knowledge they have acquired.

A **user-item filtering** takes a particular user, find users that are similar to that user based on similarity of ratings, and recommend items that those similar users liked

(focus on similarity between users to recommend items). In contrast, **item-item filtering** will take an item, find users who liked that item, and find other items that those users or similar users also liked. It takes items and outputs other items as recommendations (focus on similarity between items based on user ratings to recommend items).

b. Forklar hvordan du vil gå frem for å beregne predikert/estimert verdien av P. Gjør de antakelsene du finner nødvendige. (5%)

Svar: Here we apply item-item filtering as assumed above:

Step 1: Compute the similarities between item #6 and the rest (i.e., sim(6, i), where i = 1,2,3,4,5. Here we could use either cosinus, cosinus centered on 0, or Pearson Correlation to compute similarities.

Step 2: Find ***all positive similarities*** and compute weighted average of ratings to estimate the value of P.