Informatics 1: Data & Analysis
Session 2014/2015, Semester 2

Assignment Feedback

This is a report on the written coursework assignment for *Informatics 1: Data & Analysis.* That assignment included exam questions from 2013 and 2014, and this report is based on the solutions submitted by students taking the exams in those years. Please note the following:

- This is not a set of "model" answers. It does contain solutions, which can be used to check your own answers; but there are also extended discussions of different possible answers, key points, possible errors, and comments on the ways people approached each question on the day.
- Not all the questions have a single "right" answer. There can be multiple correct ways to write a database query, explain a concept, or construct an example. This report includes several variants on answers, but still cannot cover every possible correct alternative.
- Studying past exam questions is one way to learn more about a subject, but it is quite limited and not enough on its own. Even when an exams routinely follow a fixed structure, the questions change and successful performance does essentially depend on a good understanding of the material in the course.

The assignment consisted of three questions, each with several subquestions. The rest of this report gives the full text of each question followed by notes on solutions and feedback on the answers given by students.

Where you find errors in these notes, please send them to me at Ian.Stark@ed.ac.uk

Ian Stark
2015-03-29

Question 1 [*This question is worth a total of 40 marks.*]

A phone company wants to set up their own App Store for mobile devices. Requirements analysis for the controlling database highlights the following information about what must be recorded.

- Every app in the store needs a unique name, a publisher, and a rating.
- There are two subclasses of app: a *premium* app has to be paid for before installation; a *freemium* app is free to download, but has in-app purchases which cost money.
- The database should record the price of each premium app.
- Each user of the store is identified by their email address.
- A user may have several subaccounts, each identified by a nickname.
- The database needs to record which users have installed which apps.
- Users can use subaccounts to restrict access to freemium apps: the database needs to record which nicknames are allowed to run which ones.

(a) Draw an entity-relationship diagram to represent this information. [*20 marks*]

The app store groups apps into *themes* such as "Games", "News + Magazines", or "Health + Fitness". An app can be in multiple themes, and each theme can have a current "Top App". This is captured by the following SQL data declarations.

```
create table App (
  name      varchar(30),
  publisher varchar(25),
  rating    integer,
  primary key (name)
)
```

```
create table Theme (
  title    varchar(20),
  topApp   varchar(30),
  primary key ( title ),
  foreign key  (topApp) references App(name)
)

create table InTheme (
  name  varchar(30),
  title  varchar(20),
  primary key (name,title),
  foreign key  (name)  references App,
  foreign key  ( title )  references Theme
)
```

(b) What do the terms "arity" and "cardinality" mean when describing database tables? [*2 marks*]

(c) Write relational algebra expressions to compute the following.

(i) The name of the top app in the "Games" theme.

(ii) For every app in the "Games" theme, its name and rating. [*6 marks*]

(d) Write expressions in the tuple-relational calculus that express the following queries.

(i) The names of all apps in the "Office" theme.

(ii) The publishers of all top apps. [*6 marks*]

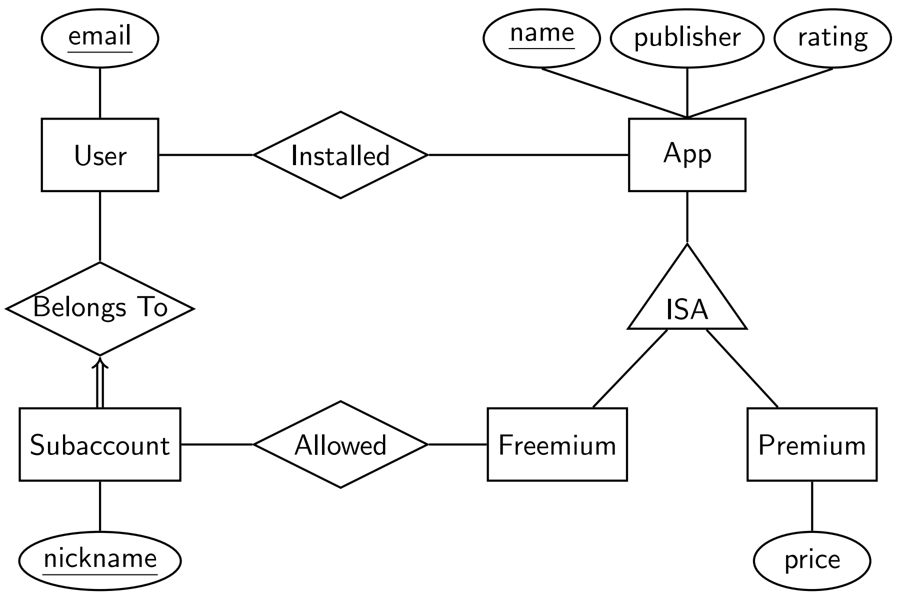
(e) Write SQL queries to answer the following questions.

(i) How many apps are there in the database?

(ii) What is the highest and lowest rating given to apps in the "Utilities" theme? [*6 marks*]

Notes on Question 1

(a) The following ER diagram captures the information required.



Notice that the ISA relationship means that Freemium and Premium app entities don't need their own copies of the name, publisher and rating attributes.

The constraints here are as follows.

- *Total participation* of Subaccount in BelongsTo, as every subaccount must belong to some user. Shown by a double (or thick) line in the diagram joining them.
- *Key constraint* between Subaccount and BelongsTo, as every subaccount must belong to at most one user. Shown by an arrowhead on the line in the diagram.

Together these capture that each subaccount belongs to exactly one user.

Some answers made Subaccount a weak entity, with Belongs To as the identifying relationship. This is a reasonable design choice: in particular, if we expect that nickname is not unique across all subaccounts of all users. To show this in the diagram, both Subaccount and BelongsTo should have a double (or thick) outline; and nickname attribute should have a double (or dotted) underline.

Note that it's not enough to just draw a double outline on Subaccount — that entity is involved in two different relationships, Belongs To and Allowed, and we have to show which one is used to uniquely identify instances of the weak entity.

Some students added publisher to the key for the App entity. This is wrong, as the scenario already states that app names are unique; and keys should be minimal.

Some students put an ISA subclass relationship between Subaccount and User. This is incorrect: a subaccount is not a special kind of user; and there can be multiple subaccounts per user. (This is a "has-a" relationship, not an "is-a" relationship.)

Some students incorrectly left out the double line between Subaccount and Belongs To. This is needed because every subaccount must correspond to some user.

Several students added an arrowhead to the line joining App to Installed. That's wrong: such a key constraint would require that each app be installed by at most one user.

Note that the scenario says "the database needs to record which nicknames are allowed to run which" freemium apps. That is why there is a relationship Allowed between Nickname and Freemium in the diagram above. Alternatives like Forbidden, Restricted, or Permission with a boolean yes/no attribute, are close but do not exactly satisfy the scenario given.

(b) The *arity* of a database table is the number of columns (fields, attributes) it has. The *cardinality* of a database table is the number of rows (tuples, records) it contains.

This is bookwork; although with variations following all the different names used to describe the columns and rows of a database table.

(c) The following relational algebra expressions compute the required sets.

(i) $\pi_{\mathsf{topapp}}(\sigma_{\mathsf{title}='\mathsf{Games}'}(\mathsf{Theme}))$

(ii) Either $\pi_{\mathsf{name,rating}}(\sigma_{\mathsf{title}='\mathsf{Games}'}(\mathsf{InTheme}) \bowtie \mathsf{App})$
or $\pi_{\mathsf{name,rating}}(\sigma_{\mathsf{title}='\mathsf{Games}'}(\mathsf{InTheme} \bowtie \mathsf{App}))$ will do.

Note that because InTheme and App share the foreign key name we can use a natural join "$\bowtie$".

Some students explicitly described the join, with "$\bowtie_{\mathsf{Intheme.name=App.name}}$"; that's also correct.

Taking the join Theme $\bowtie$ App is incorrect, as these tables do not share any fields.

(d) The queries given can be captured by these tuple-relational calculus expressions.

(i) $\{ R \mid \exists X \in \mathsf{InTheme}\ .\ X.\mathsf{title} = '\mathsf{Office}' \wedge X.\mathsf{name} = R.\mathsf{name} \}$

(ii) $\{ R \mid \exists T \in \mathsf{Theme}, A \in \mathsf{App}\ .\ T.\mathsf{topApp} = A.\mathsf{name} \wedge A.\mathsf{publisher} = R.\mathsf{publisher} \}$

In both cases an auxiliary relation R is needed to pick out the single desired field.

Note that a tuple-relational calculus expression returns a set of records, not individual fields. Expressions beginning $\{ X.\mathsf{name} \mid \ldots \}$ or similar are incorrect, and not meaningful in TRC. Use an auxiliary relation like R instead.

(e) SQL:

(i) **select count**(∗) **from** App
or **select count**(name) **from** App is fine too.
There is no need to add **distinct**, because each name is already unique.

(ii) **select min**(rating), **max**(rating)
from App, InTheme
where App.name = InTheme.name **and** InTheme.title = 'Utilities'

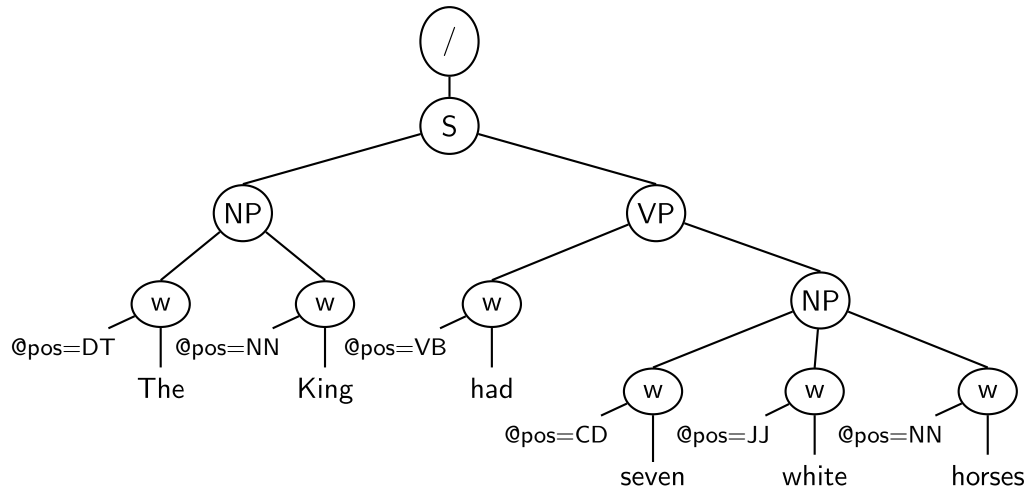
It's possible, but not necessary, to use an explicit join for this.

select min(rating), **max**(rating)
from App **join** InTheme **on** App.name = InTheme.name
where InTheme.title = 'Utilities'

Any SQL query engine should compile both of these to exactly the same execution plan.

Here **count**, **min** and **max** are *aggregate operations* of SQL. These go beyond relational algebra and tuple-relational calculus to compute summary values over the multisets that are returned by basic queries.

Question 2 [*This question is worth a total of 30 marks.*]



(a) The tree above is the XPath data model for an annotated parse of the sentence "The King had seven white horses", using the following abbreviations for syntactic components and part-of-speech tags:

S	Sentence	DT	Determiner (e.g. the, a, each)
NP	Noun phrase	NN	Noun (e.g. King, horse)
VP	Verb phrase	VB	Verb (e.g. has, lives)
w	Word	CD	Cardinal number (e.g. seven)
pos	Part of speech	JJ	Adjective (e.g. large, white)
CC	Conjunction	IN	Preposition (e.g. for, or, in)

Write out the XML text form for this document. [*12 marks*]

(b) The following passage has been marked up with parts of speech, as in the Corpus Workbench used with the Corpus Query Processor (CQP).

```
The/DT proud/JJ King/NN of/IN distant/JJ Spain/NN had/VB seven/CD beautiful/JJ
white/JJ horses/NN . An/DT evil/JJ wizard/NN cursed/VB the/DT King/NN and/CC
stole/VB the/DT horses/NN .
```

A *noun phrase* is a phrase of one or more words that taken together play the grammatical role of a noun. For example, this passage includes the following noun phrases:

- The proud King of distant Spain
- The proud King
- distant Spain
- seven beautiful white horses
- An evil wizard
- the King
- the horses

In the CQP syntax an expression like [**pos**="NN"] matches a single noun. Write CQP regular expressions to match the following:

(i) A sequence of two or more adjectives in a row.
(ii) A sequence of words, each of which is either a noun or a verb.
(iii) All of the noun phrases given above. [*12 marks*]

(c) A large research corpus such as the *British National Corpus* or the *Corpus of Contemporary American English* contains hundreds of millions of words from many different sources. Building such a corpus requires balancing and sampling to ensure it is representative.

Explain briefly the meaning of *balancing*, *sampling* and *representative* as used here. [*6 marks*]

Notes on Question 2

(a) Here is the sentence as an XML document.

```
<?xml version="1.0" encoding="utf−8"?>
<!DOCTYPE S SYSTEM "sentence.dtd">
<S>
  <NP><w pos="DT">The</w><w pos="NN">King</w></NP>
  <VP>
    <w pos="VB">had</w>
    <NP>
      <w pos="CD">seven</w>
      <w pos="JJ">white</w>
      <w pos="NN">horses</w>
    </NP>
  </VP>
</S>
```

The opening <?**xml** ...?> clause is required in a well-formed XML document, but wasn't marked down if absent in this case. Similarly, I've included a <!**DOCTYPE** ...> declaration, but that's not essential.

The key requirements here are to correctly capture the nested element structure, the content of the text nodes, and the part-of-speech attributes.

Several students used tags like <Sentence> and <Verb Phrase>, or some other variation on those given in the tree. This is incorrect: each XPath tree has a precisely equivalent XML document, using exactly the same element names and attributes.

(b) Each of these have multiple possible representations as regular expressions. Here are just a few.

(i) [**pos**="JJ"] [**pos**="JJ"]+
or [**pos**="JJ"] [**pos**="JJ"] [**pos**="JJ"]∗
Note that the part of speech is "JJ" not "JJ.∗". That's because this question specifically lists the part-of-speech tags used, and the only adjective tag is "JJ". This is different from the larger tag sets used in examples like the Dickens corpus from tutorials.

(ii) ([**pos**="NN"] | [**pos**="VB"])+
or [**pos**="NN|VB"]+
or ([**pos**="NN"]∗ [**pos**="VB"]∗)∗ (although this matches the empty sequence)

(iii) [**pos**="DT"]?[**pos**="CD"]?[**pos**="JJ"]∗[**pos**="NN"]
([**pos**="IN"][**pos**="DT"]?[**pos**="CD"]?[**pos**="JJ"]∗[**pos**="NN"])?

There are also other possibilities using some of the additional operations provided by CQP on regular expressions, such as using braces "$\{n,m\}$" to indicate repetition between n and m times.

These operations are not a required part of the course, but acceptable in the exam if used correctly. Note that availability of these more advanced features can vary significantly between different tools. You can usually expect the operators described in the course ("∗", "+", "?", "|") to be available in all situations and with the same meanings; for everything else, check the manual.

(c) **Balancing** means choosing a range of different types of sources for the corpus: books, newspapers, blogs, letters, etc.

Sampling refers to selecting texts at random from the chosen sources.

Representative A corpus is *representative* if it contains a similar mix of text to the language variant for which it is being developed.

This is bookwork: recall of material exactly as described in the lecture course. However, it's still easier to remember and present clearly if you do understand what the terms mean, and their motivation. For example, many students confused balancing and sampling — the difference is that balancing is done to give variety to the corpus, and sampling is there to remove bias in selection.

One common type of error was to explain why something is done, rather than what it is. For example "balancing is done to ensure the corpus is representative". That's simply restating the question; an answer should explain what balancing is, and possibly how it is done.

Question 3 [*This question is worth a total of 30 marks.*]

(a) An information retrieval system is searching a European Parliament archive for documents on the topic of "offshore fishing boundary disputes". The following document matrix indicate three possible matches.

	offshore	fishing	boundary	disputes
Document A	4	2	7	0
Document B	3	3	3	3
Document C	12	6	0	0
Query	1	1	1	1

One way to rank these documents for potential relevance to the topic is the *cosine similarity measure.*

Write out the formula for calculating the cosine of the angle between two four-dimensional vectors (x_1, x_2, x_3, x_4) and (y_1, y_2, y_3, y_4).

Use this to rank the three documents in order of relevance to the query. [*10 marks*]

(b) One way to evaluate the performance of an information retrieval system is to assess its *precision* P and *recall* R. Informally, P can be defined as the proportion of the documents returned by the system which do match the objectives of the original search. Give a similar informal definition of R.

Here is the mathematical formula for calculating precision.

$$P = \frac{TP}{TP + FP}$$

Name and define the terms TP and FP here. Give the formula for recall R, explaining any other new terms that appear. [*8 marks*]

(c) You have been given two different information retrieval systems to compare: *Hare* and *Tortoise.* Each one is tested on the same query for a collection of 4000 documents, of which 200 are relevant to the query. *Hare* returns 1200 documents, including 150 that are relevant; while *Tortoise* returns just 160, with 120 of them being relevant.

Tabulate the results for each system and calculate their precision and recall on this test. Show your working.

One way to combine precision and recall scores is to use their *harmonic mean.* Give the formula for this, and calculate its value for each of *Hare* and *Tortoise.* [*12 marks*]

Notes on Question 3

(a) The cosine formula for four-vectors is:

$$\cos(\vec{x}, \vec{y}) = \frac{x_1y_1 + x_2y_2 + x_3y_3 + x_4y_4}{\sqrt{x_1^2 + x_2^2 + x_3^2 + x_4^2}\sqrt{y_1^2 + y_2^2 + y_3^2 + y_4^2}}$$

This explicitly uses the coordinate values provided in the question.

There is an alternative presentation using only vector notation:

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \,.\, \vec{y}}{|\vec{x}||\vec{y}|}$$

However, this doesn't precisely answer the question, which specifically requests a formula given the coordinates (x_1, x_2, x_3, x_4) and (y_1, y_2, y_3, y_4).

To rank the three documents, use this formula to calculate the cosine between each document and the original query.

$$\begin{aligned}
\cos(\text{Document A}, \text{Query}) &= \frac{4+2+7}{\sqrt{4}\sqrt{4^2+2^2+7^2}} &= 0.78\\
\cos(\text{Document B}, \text{Query}) &= \frac{3+3+3+3}{\sqrt{4}\sqrt{3^2+3^2+3^2+3^2}} &= 1\\
\cos(\text{Document C}, \text{Query}) &= \frac{12+6+0}{\sqrt{4}\sqrt{12^2+6^2}} &= 0.67
\end{aligned}$$

This ranks the three documents in order as:

- Document B
- Document A
- Document C

The largest cosine value indicates the document most closely matching the query.

Some students identified only the leading document, B. That's not enough to answer the question, which asked for a ranking of all three documents.

A few students referred throughout to documents 1, 2 and 3; this is incorrect, as they don't appear in the question at all.

While it's possible to go on and compute the angle between the each document vector and the query, it's not necessary. For cosine similarity, you only need to rank the cosine values in order.

Several people instead calculated the cosine between pairs of documents: A and B, B and C, C and A. That's incorrect, and definitely doesn't help rank them by relevance to the query: the "similarity" part of cosine similarity is between each document and the query.

(b) The *recall* R is the proportion of relevant documents in the collection which are successfully retrieved.

As with "balancing" and "sampling" in Question 2, it's important here to be able to precisely express what a technical term like "recall" signifies, as well as having a general understanding of the context. For example, notice that both precision and recall are a *proportion* of documents (which will be a real number between 0 and 1), not the total number (which will be some non-negative integer).

Names and definitions of terms:

- *TP* is *True Positives*, the number of relevant documents correctly returned.
- *FP* is *False Positives*, the number of irrelevant documents returned.

Notice that these terms are referring to whole numbers, not proportions.

The formula for calculating recall R is

$$R = \frac{TP}{TP + FN}$$

where FN is *False Negatives*, the number of relevant documents incorrectly rejected.

Because the question asks you to explain any new terms that appear in the formula for R, it's important to include both the term "False Negatives" and a description of what it means.

(c) The following tables give all the necessary figures for the calculation.

Hare	Relevant	Not relevant	Total
Retrieved	150	1050	1200
Not retrieved	50	2750	2800
Total	200	3800	4000

Tortoise	Relevant	Not relevant	Total
Retrieved	120	40	160
Not retrieved	80	3760	3840
Total	200	3800	4000

From these we can evaluate the performance of each system on this single query.

$$\textit{Hare} \text{ precision } P = \frac{150}{1200} = 1/8 = 0.125 \qquad \textit{Tortoise} \text{ precision } P = \frac{120}{160} = 3/4 = 0.75$$

$$\textit{Hare} \text{ recall } R = \frac{150}{200} = 3/4 = 0.75 \qquad \textit{Tortoise} \text{ recall } R = \frac{120}{200} = 3/5 = 0.6$$

This answers the question, which asks "Tabulate ... calculate ... Show your working". In particular, "tabulate" means to write out a table — so you need a table. Here, this *contingency table* with row and column totals makes it straightforward to calculate the answers: for each of them we need only pick out two values from the table and divide one by the other.

Some people used a different table showing "True" and "False" against "Positive" and "Negative". That turns out not to be so useful in calculating precision and recall. It's possible, but a bit more awkward as you'll find you need to add up a pair of diagonal elements rather than just picking out single values. I don't recommend this.

The formula for calculating the harmonic mean has many equivalent presentations. One of the simplest is

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

but you could also use

$$H = \frac{1}{\frac{1}{2}\frac{1}{P} + \frac{1}{2}\frac{1}{R}} \quad \text{or} \quad H = \frac{2}{\frac{1}{P} + \frac{1}{R}} \quad \text{or} \quad \frac{1}{H} = \frac{1}{2}\left(\frac{1}{P} + \frac{1}{R}\right).$$

Using these we can calculate the harmonic mean of precision and recall for *Hare* as $3/14 = 0.21$ and for *Tortoise* as $2/3 = 0.67$.