0007 sheet 1 učo 409729 points

Describe the SoundEx phonetic retrieval algorithm [2 points]. Give an example of two similarly-sounding words with the same SoundEx code [1 point]. Explain the weak point of SoundEx [1 point], and give an example of two similarly-sounding words with different SoundEx codes [1 point].

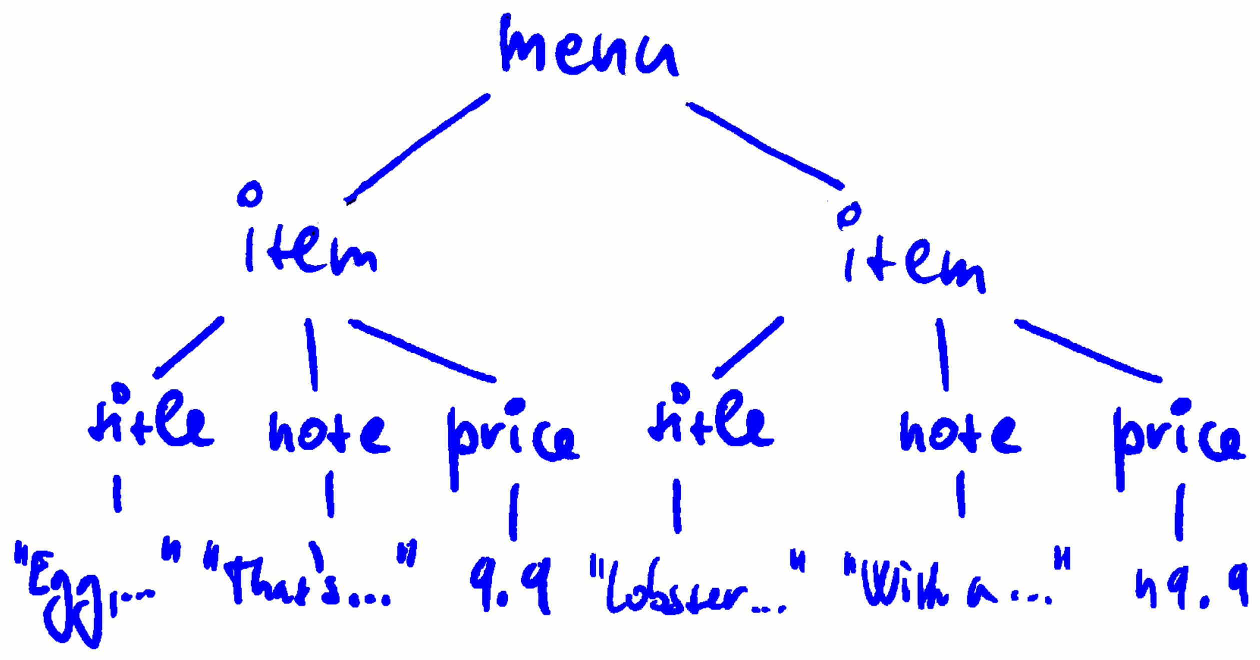
Generally speaking, the SoundEx algorithm maps similarly-sounding words together by removing consonants, and clustering vowels with similar English spelling together.

Sword and short have code S63∅.

The major weak point of SoundEx is the fact that the initial letter of a word is retained — fog is F2∅∅, whereas thug is T2∅∅.

0007 sheet 2 učo 409729 points

Consider the following XML document:




```
<?xml version="1.0" encoding="UTF-8"?>
<menu>
    <item>
        <title>Egg, sausage, and spam</title>
        <note>That's not got much spam in it</note>
        <price unit="GBP">9.9</price>
    </item>
    <item>
        <title>Lobster Thermidor</title>
        <note>With a fried egg on top, and spam</note>
        <price unit="GBP">49.9</price>
    </item>
</menu>
```

Draw the XML document as a graph [1 point], and count all *structural terms* in the document [2 points]. Compute the *structural similarity* between the structural term `item/title#"That's not got much spam in it"`, and the queries `//menu//price#30.9`, and `//title#"Lobster Thermidor"` [2 points].

Structural terms: #"Egg, ...", title#"Egg, ...", item/title#"Egg, ...", menu/item/title#"Egg, ...", #"That's...", note#"that's...", item/note#"That's...", menu/item/note#"That's...", #9.9, price#9.9, item/price#9.9, menu/item/price#9.9, #"Lobster...", title#"Lobster...", item/title#"Lobster...", menu/item/title#"Lobster...", #"With a...", note#"With a...", item/note#"With a...", menu/item/note#"With a...", #49.9, price#49.9, item/price#49.4, menu/item/price#49.9

$6 \cdot 4 = 24$ structural terms.

//menu//price does not expand to item/title, the structural similarity is zero. //title expands to item/title by adding a single path segment, the structural similarity is $\frac{1+1}{1+2} = \frac{2}{3}$.

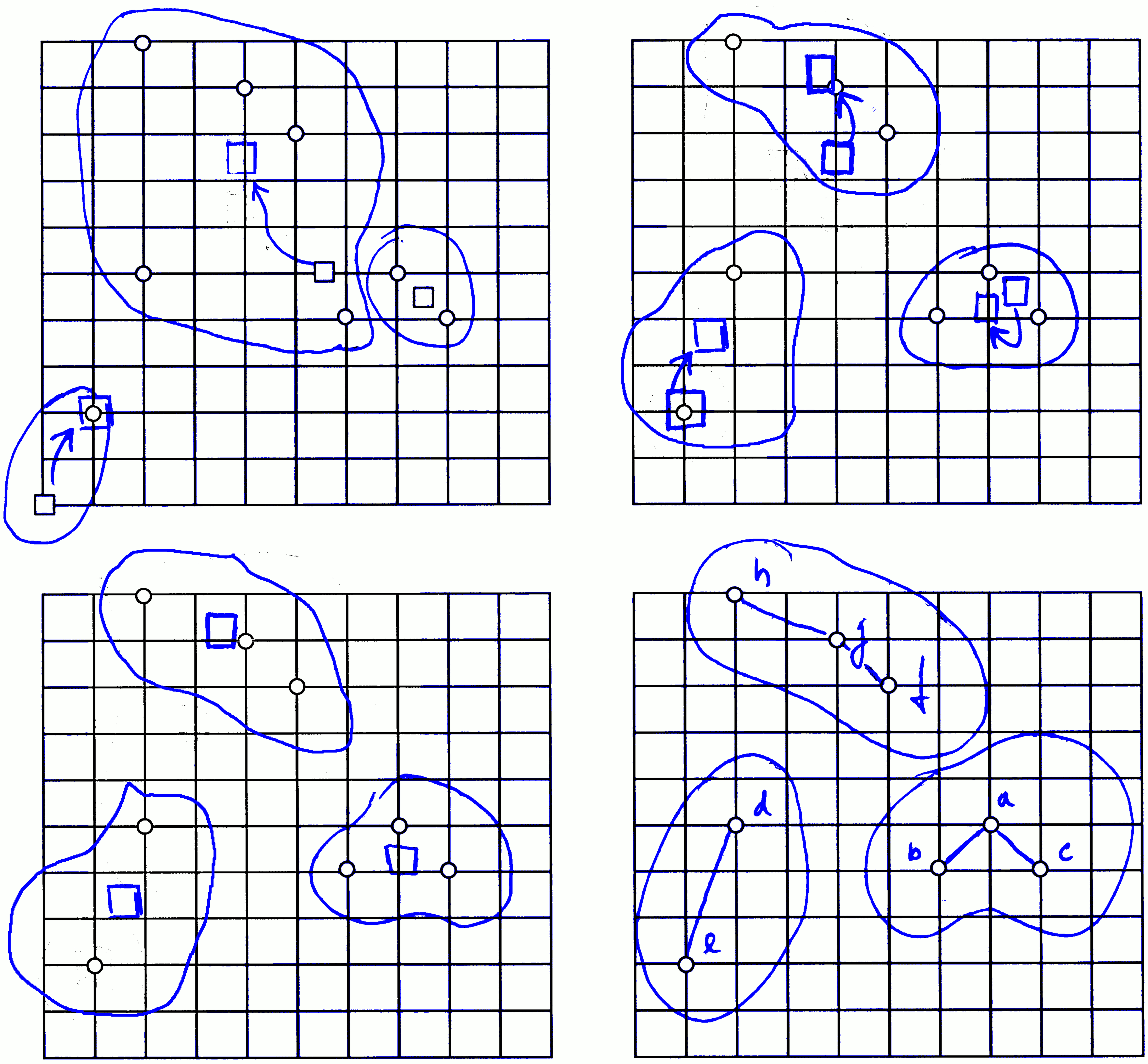
0007 sheet 3 učo 409729 points

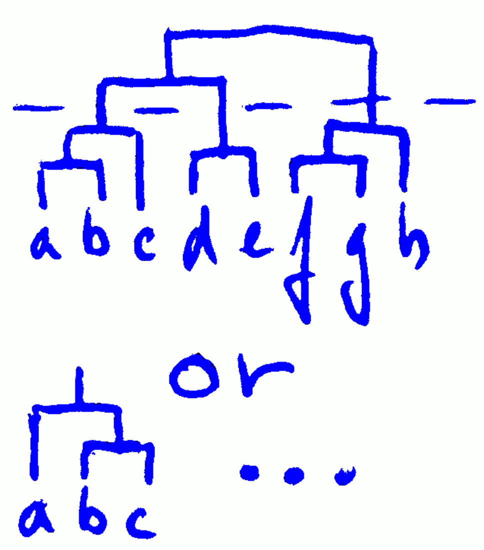
Explain the following aspects of the K-means flat clustering algorithm [2 points]:

1. What do we need to know about our dataset before using the algorithm?
2. What is the input and the output of the algorithm?
3. What are the two steps that take place in every epoch?
4. How do we decide in which epoch to stop the algorithm?

1. We need to know the number of classes K and initial mean estimates (seeds). 2. The input are unclassified points and K seeds. 3. Reassigning points, recomputing centroids. 4. Centroids converged.

Given the points ○, and the seeds □, run the K-means algorithm for three epochs. Draw the state of the algorithm at the beginning and after every epoch; no computation should be necessary. What is the output of the algorithm? [2 points]





Centroids converged. Hierarchical clustering.

Perform a hierarchical clustering of the above dataset into three classes using the single-link hierarchical agglomerative clustering algorithm, and draw the resulting *dendrogram*. [1 point] Is the output the same as the output of the K-means flat hierarchical clustering algorithm above? [1 point]

Yes, it is.

You may also write on the other side of the sheet, but it won't be scanned.

0007 sheet 4 učo 409729 points

State the *Zipf's law* [1 point] and the *Heaps' law* [1 point] in its complete variant [1 point]. Let C denote the *term-document matrix* of a document collection that contains M unique terms. According to the Heaps' law, what is the size of C in relation to M? [3 points]

Zipf's law: $cf_i = c/i$, where cf_i is the collection frequency of the ith most frequent term, and c is a constant.

Heaps' law: $M = kT^b$, where M is the vocabulary size, T is the collection size in tokens, and $30 \leq k \leq 100$ and $b \approx 0{,}5$ are parameters.

The size of **C** is $M \times N < T$, where N is the number of documents in a collection. According to the Heaps' law, $T = \sqrt[b]{M/k} \approx M^2$.

You may also write on the other side of the sheet, but it won't be scanned.

0007 sheet 5 učo 409729 points

Consider the following collection of four documents d_i:

- d_1: HOME SALES RISE IN JULY
- d_2: JULY NEW HOME SALES RISE
- d_3: NEW HOME SALES TOP FORECASTS
- d_4: INCREASE IN HOME SALES IN JULY

Produce a list of (term, document ID) tuples [1 point], sort this list in lexicographical order [1 points], and use the sorted list to construct an inverted index [3 points]. Write down each step. Describe how you would produce this index using the MapReduce distributed framework [3 points].

(home, 1), (sales, 1), (rise, 1), (in, 1), (July, 1), (July, 2), (new, 2), (home, 2), (sales, 2), (rise, 2), (new, 3), (home, 3), (sales, 3), (top, 3), (forecasts, 3), (increase, 4), (in, 4), (home, 4), (sales, 4), (in, 4), (July, 4)

(forecasts, 3), (home, 1), (home, 2), (home, 3), (home, 4), (in, 1), (in, 4), (increase, 4), (July, 1), (July, 2), (July, 4), (new, 2), (new, 3), (rise, 1), (rise, 2), (sales, 1), (sales, 2), (sales, 3), (sales, 4), (top, 3)

forecasts -> 3
home -> 1 -> 2 -> 3 -> 4
in -> 1 -> 4
increase -> 4
new -> 2 -> 3
July -> 1 -> 2 -> 4
rise -> 1 -> 2
sales -> 1 -> 2 -> 3 -> 4
top -> 3

Each parser would process a limited number of documents and produce a sorted (term, document ID) list. Each inverter would take all sublists in a certain alphabetical range of terms and produce postings for that alphabetical range.

You may also write on the other side of the sheet, but it won't be scanned.