UNIVERSITY OF MALAYA

WQD7007 Big Data Management

Big Data Concepts

Providing Structure to Unstructured Data

Previous lecture

- What is Hadoop
- Big data pipeline using Hadoop

Agenda

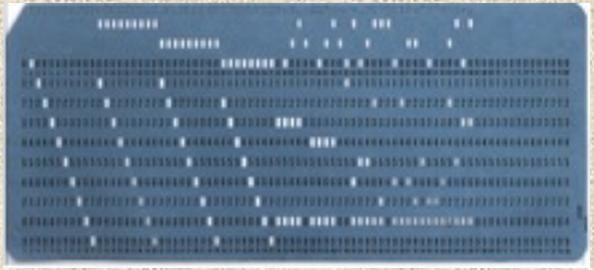
- Providing structure to unstructured data
    - Machine translation
    - Autocoding
    - Indexing
    - Term Extraction

Providing structure to unstructured data

- In the early days of computing, data was always highly structured.
  - E.g. **All data was divided into fields**
    - the fields had a fixed length, and the data entered into each field was constrained to a predetermined set of allowed values.
  - Data was entered into **punch cards**, with preconfigured rows and columns.
    - Depending on the intended use of the cards, **various entry and read-out methods were chosen to express binary data, numeric data, fixed-size text, or programming instructions**
    - For many **analytic purposes**, card-encoded data sets were analyzed without the assistance of a computer; all that was needed was a punch card sorter.

Providing structure to unstructured data

- Punch cards:



- Punch card sorters:

- In short, all data are expected to be structured data in early days of computing.

Providing structure to unstructured data

- Today, **most data entered by humans is unstructured**, in the form of free text.
  - E.g. e-mail messages, tweets, documents, and so on.

- Structured data has not disappeared, but it sits in the shadows cast by mountains of unstructured text.
  - Free text may be more interesting to read than punch cards, but punch cards was much easier to analyze.

Providing structure to unstructured data

- To get much informational value from free text, it is necessary **to impose some structure**. This may involve:
  - translating the text to a preferred language
  - parsing the text into sentences
  - extracting and normalizing the conceptual terms contained in the sentences
  - mapping terms to a standard nomenclature annotating the terms with codes from one or more standard nomenclatures
- All of these activities are difficult to do on a small scale and virtually impossible to do on a large scale
  - Every big data project that uses unstructured data must deal with these tasks to yield the best possible results with the resources available.

Machine Translation

- Unstructured data: data objects whose contents are **NOT** organized into arrays of attributes or values
  - Example: paragraph in a book
- Structured data?
  - Example: Spreadsheets, with data distributed in cells, marked by a row and column position
- **Without structure**, the contents of the data cannot be **sensibly collected and analyzed**.
  - Because Big Data is immense, the tasks of imposing structure on text must be **automated and fast**.

Machine Translation

- What is machine translation?
  - Translate text **from one language into another language**.
  - is one of the better known areas in which computational methods have been applied to free text.
- **Process**:
  1. extracting sentences from text
  2. parsing the words of the sentence into grammatical parts
  3. arranging the grammatical parts into an order that imposes logical sense on the sentence.

Machine Translation

- Next, each of the parts **can be translated by a dictionary** that finds **equivalent terms in a foreign language** to be reassembled by applying **grammatical positioning rules** appropriate for the **target language**.
  - Because this process uses the natural rules for sentence constructions in a foreign language, the process is often referred to as natural language machine translation.

- Example?

Machine Translation

- **Pros**: It all seems simple and straightforward if **proper look-up tables are developed**.

- **Cons**: Languages do not make much sense for computers.
  - Computers cannot find meaning in sentences **that have no meaning**.
  - humans find meaning in the English language because we **impose our own cultural background** onto the sentences we read, to create meaning where none exists.

Autocoding

- Coding, as used in the context of unstructured textual data, is the process of **tagging terms with an identifier code** that corresponds to a **synonymous term** listed in a standard nomenclature
  - Nomenclature: the devising or choosing of names for things, especially in a science or other discipline
  - Example: a medical nomenclature might contain the term *renal cell carcinoma*, a type of kidney cancer, attaching a unique identifier code for the term, such as “C9385000.”

Autocoding

- There are about **50 recognized synonyms** for "*renal cell carcinoma*."
  - A few of these synonyms and near-synonyms are listed here to show that a single concept can be expressed many different ways, including
    - adenocarcinoma arising from kidney,
    - adenocarcinoma involving kidney,
    - cancer arising from kidney,
    - carcinoma of kidney,
    - Grawitz tumor,
    - Grawitz tumour,
    - hypernephroid tumor,
    - hypernephroma,
    - kidney adenocarcinoma,
    - renal adenocarcinoma,
    - renal cell carcinoma.
- All of these terms could be assigned the same identifier code, "C9385000."

Autocoding

- The process of coding a text document involves **finding all the terms** that belong to a **specific nomenclature** and **tagging** the term with the corresponding **identifier code**.

- Example, there may be a nomenclature of diseases, or makes and models of automobiles.
  - Some nomenclatures are ordered alphabetically.
  - Others are ordered by synonymy, wherein all synonyms and plesionyms are collected under a canonical (i.e., best or preferred) term.

Autocoding

- Purpose of Nomenclatures includes:
  - to enhance interoperability and integration,
  - to allow synonymous terms to be retrieved regardless of which specific synonym is entered as a query,
  - to support comprehensive analyses of textual data

- Traditionally, nomenclature coding has been **considered a specialized and highly detailed task** that is best accomplished by human beings.
  - Tagging documents with nomenclature codes is serious business. If the coding is flawed, the consequences can be dire.

Autocoding

- Example: In 2009, the Department of Veterans Affairs (VA) sent out hundreds of letters to veterans with the devastating news that they had contracted amyotrophic lateral sclerosis, also known as Lou Gehrig’s disease, a fatal degenerative neurologic condition.
  - About 600 of the recipients did not, in fact, have the disease. The VA retracted these letters, attributing the confusion to a coding error.
  - Coding text is difficult. **Human coders are inconsistent, idiosyncratic, and prone to error.**

Autocoding

- When dealing with text in gigabyte and greater quantities, human coding is simply out of the question.
    - **There is not enough time, or money, or talent to manually code the textual data contained in** Big Data resources.
    - Computerized coding (i.e., autocoding) is the only practical solution.

- **Cons**: The best natural language autocoders are **pitifully slow.** The reason for the slowness relates to their algorithm
    - As a result, a good natural language autocoder parses text at about 1 kilobyte per second.

Autocoding

- The algorithm steps:
  1. parsing text into sentences;
  2. parsing sentences into grammatical units,
  3. rearranging the units of the sentence into grammatically permissible combinations,
  4. expanding the combinations based on stem forms of words,
  5. allowing for singularities and pluralities of words,
  6. matching the allowable variations against the terms listed in the nomenclature.
- This means that if an autocoder must parse and code **a terabyte of textual material**, it would require 1000 million seconds to execute, or about **30 years**.
- Big Data resources typically contain many terabytes of data; thus, natural language autocoding software is unsuitable for translating Big Data resources.

Autocoding

- This being the case, what good are they? (**Pros**)
- Natural language autocoders have value when they are employed at the time of **data entry**.
    - **Humans type sentences at a rate far less than 1 kilobyte per second**, and natural language autocoders can keep up with typists, inserting codes for terms, as they are typed.
        - They can operate much the same way as auto**correct, autospelling, look-ahead**, and other commonly available crutches intended to **improve or augment the output** of plodding human typists.
        - In cases where a variant term evades capture by the natural language algorithm (e.g. unrecognizable grammatical structure), the typist can supply the application with an equivalent (i.e., renal cell carcinoma = rcc) that can be stored by the application and applied against future inclusions of alternate forms.

Indexing

- A well-designed book index is a creative, literary work that **captures the content and intent of the book** and transforms it into a listing wherein related concepts, found scattered throughout the text, are **collected under common terms and keyed to their locations**.
  - to create a keyed encapsulation of concepts, subconcepts, and term relationships.

Indexing

- Big Data resources that lack a proper index **CANNOT** be utilized to their full potential. **Without an index, you never know what your queries are missing.**
  - it is the **relationship among data objects that are the keys to knowledge.** Data by itself, even in large quantities, tells only part of a story.
  - electronic indexes map concepts, classes, and terms to specific locations in the resource where data items are stored
  - An index imposes **order and simplicity** on the big data resource.

Indexing

- Indexing? Why not just use the ”find” function?

1. An index can be read, like a book, to acquire a **quick understanding of the contents and general organization** of the data resource.

2. When you do a “find” search in a query box, your search may come up empty if there is nothing in the text that matches your query.
   - This can be very frustrating if you know that the text covers the topic entered into the query box. **Indexes avoid the problem of fruitless searches.**
   - By browsing the index you can find the term you need, without foreknowledge of its exact wording within the text.

Indexing

3. Index searches are instantaneous, even when the Big Data resource is enormous.
   - Indexes are constructed to contain the results of the search of every included term, **obviating the need to repeat the computational task of searching** on indexed entries.

4. Indexes can be tied to a classification.
   - This permits the analyst to know the **relationships among different topics within the index** and within the text.

5. Many indexes are **cross-indexed**, providing relationships among index terms that might be extremely helpful to the data analyst.

Term Extraction

- Consider the following: “The diagnosis is chronic viral hepatitis.”
  - This sentence contains two very **specific medical concepts**: “diagnosis” and “chronic viral hepatitis.”
  - These two concepts are connected to form a meaningful statement with the words “the” and “is,” and the sentence delimiter, “.” “The,” “diagnosis,” “is,” “chronic viral hepatitis,” “.”
  - **A term** can be defined as a **sequence** of one or more uncommon words that are demarcated (i.e., bounded on one side or another) by the occurrence of one or more common words, such as “is,” “and,” “with,” “the.”

Term Extraction

- If we had **a list of all the words that were considered common**, we could write a program that extracts all the concepts found in any text of any length.
  - The concept terms would consist of all **sequences of uncommon words that are uninterrupted by common words.**

Term Extraction

- An algorithm for extracting terms from a sentence follows:
  1. Read the first word of the sentence. If it is a common word, delete it. If it is an uncommon word, save it.
  2. Read the next word.
     1. If it is a common word, delete it and place the saved word (from the prior step, if the prior step saved a word) into our list of terms found in the text.
     2. If it is an uncommon word, append it to the word we saved in step one and save the two-word term.
     3. If it is a sentence delimiter, place any saved term into our list of terms and stop the program.
  3. Repeat step two.

  "diagnosis" "chronic viral hepatitis"

Term Extraction

- This simple algorithm, is a fast and efficient method to build a collection of **index terms**.
    - To use the algorithm, you must prepare or find a list of common words appropriate to the information domain of your big data resource.
    - Example: To extract terms from the National Library of Medicine's citation resource (about 20 million collected journal articles), the following list of common words is used:
        - "about, again, all, almost, also, although, always, among, an, and, another, any, are, as, at, be, because, been, before, being, between, both, but, by, can, could, did, do, does, done, due, during, each, either, enough, especially, etc, for, found, from, further, had, has, have, having, here, how, however, i, if, in, into, is, it, its, itself, just, kg, km, made, mainly, make, may, mg, might, ml, mm, most, mostly, must, nearly, neither, no, nor, obtained, of, often, on, our, overall, perhaps, pmid, quite, rather, really, regarding, seem, seen, several, should, show, showed, shown, shows, significantly, since, so, some, such, than, that, the, their, theirs, them, then, there, therefore, these, they, this, those, through, thus, to, upon, use, used, using, various, very, was, we, were, what, when, which, while, with, within, without, would."
    - Such lists of common words are sometimes referred to as **"stop word lists"** or **"barrier word lists,"** as they demarcate the beginnings and endings of extraction terms.

Term Extraction

- Notice that the algorithm parses through text **sentence by sentence.**
  - This is a somewhat awkward method for a computer to follow, as **most programming languages automatically cut text from a file line by line** (i.e., breaking text at the newline terminator).
  - A computer program has no way of knowing where a sentence begins or ends, **unless the programmer finds sentences, as a program subroutine.**

Term Extraction

- There are many **strategies** for determining where one sentence stops and another begins.
  - The easiest method looks for the occurrence of a **sentence delimiter** immediately following a lowercase alphabetic letter, that precedes one or two space characters, that precede an uppercase alphabetic character.

Term Extraction

- Here is an example: **“I like pizza. Pizza likes me.”**
  - Between the two sentences is the sequence **“a. P,”** which consists of a lowercase “a” followed by a period, followed by a space, followed by an uppercase “P”.
  - This general pattern (lowercase, period, one or two spaces, uppercase) usually signifies a **sentence break**.
  - The routine fails with sentences that break at the end of a line or at the last sentence of a paragraph.

Concluding remarks

How to build a good index?

1. Should the index be devoted to a **particular knowledge domain?**

   - You may want to create an index of names of persons, an index of geographic locations, or an index of types of transactions.

   - Your choice depends on the intended uses of the Big Data resource.

Concluding remarks

How to build a good index?

1. Should the index be devoted to a **particular knowledge domain?**
2. Should the index be devoted to a **particular nomenclature?**

- A coded nomenclature might facilitate the construction of an index if synonymous index terms are attached to their shared nomenclature code.

Concluding remarks

How to build a good index?

1. Should the index be devoted to a **particular knowledge domain?**
2. Should the index be devoted to a **particular nomenclature?**
3. Should the index be **built upon a scaffold that consists of a classification?**

- For example, an index prepared for biologists might be keyed to the classification of living organisms.

- Gene data has been indexed to a gene ontology and used as a research tool.

Concluding remarks

How to build a good index?

1. Should the index be devoted to a **particular knowledge domain?**
2. Should the index be devoted to a **particular nomenclature?**
3. Should the index be **built upon a scaffold that consists of a classification?**
4. In the absence of a classification, might **proximity among terms** be included in the index?
   - Term associations leading to useful discoveries can sometimes be found by collecting the distances between indexed terms.
   - Terms that are proximate to one another (i.e., co-occurring terms) tend to have a relational correspondence.
   - For example, if "aniline dye industry" co-occurs often with the seemingly unrelated term "bladder cancer," then you might start to ask whether aniline dyes can cause bladder cancer.

Concluding remarks

How to build a good index?

1. Should the index be devoted to a **particular knowledge domain?**
2. Should the index be devoted to a **particular nomenclature?**
3. Should the index be **built upon a scaffold that consists of a classification?**
4. In the absence of a classification, might **proximity among terms** be included in the index?
5. Should **multiple indexes** be created?
   - Specialized indexes might be created for data analysts who have varied research agendas.

Concluding remarks

How to build a good index?

1. Should the index be devoted to a **particular knowledge domain?**
2. Should the index be devoted to a **particular nomenclature?**
3. Should the index be **built upon a scaffold that consists of a classification?**
4. In the absence of a classification, might **proximity among terms** be included in the index?
5. Should **multiple indexes** be created?
6. Should the index be **merged into another index**?
    - It is far easier to merge indexes than to merge Big Data resources.
    - It is worthwhile to note that the greatest value of Big Data comes from finding relationships among disparate collections of data.