# A Primer on Text-Data Analysis



**Imad Rahal**
*College of Saint Benedict & Saint John's University, USA*

**Baoying Wang**
*Waynesburg College, USA*

**James Schnepf**
*College of Saint Benedict & Saint John's University, USA*

## INTRODUCTION

Since the invention of the printing press, text has been the predominate mode for collecting, storing and disseminating a vast, rich range of information. With the unprecedented increase of electronic storage and dissemination, document collections have grown rapidly, increasing the need to manage and analyze this form of data in spite of its unstructured or semistructured form. **Text-data analysis** (Hearst, 1999) has emerged as an interdisciplinary research area forming a junction of a number of older fields like machine learning, natural language processing, and information retrieval (Grobelnik, Mladenic, & Milic-Frayling, 2000). It is sometimes viewed as an adapted form of a very similar research field that has also emerged recently, namely, data mining, which focuses primarily on structured data mostly represented in relational tables or multidimensional cubes.

This article provides an overview of the various research directions in text-data analysis. After the "Introduction," the "Background" section provides a description of a ubiquitous text-data representation model along with preprocessing steps employed for achieving better text-data representations and applications. The focal section, "Text-Data Analysis," presents a detailed treatment of various text-data analysis subprocesses such as ***information extraction***, ***information retrieval*** and ***information filtering***, ***document clustering*** and ***document categorization***. The article closes with a "Future Trends" section followed by a "Conclusion" section.

## BACKGROUND

Text-data analysis is defined as the computerized process of automatically extracting useful knowledge from enormous collections of natural text documents (a.k.a. document collections) usually coming from various dynamic sources. It is a broad process embedding a number of subprocesses, all of which deal with textual resources which are naturally unstructured or semistructured, as in the case of HTML (HyperText Markup Language) and XML (eXtensible Markup Language) documents; a fact that makes it extremely difficult to apply computational solutions to real life text-based problems.

In order to alleviate the difficulty faced by computers when dealing with the unstructured nature of text-data resources, a process called *indexing* is utilized. This process is normally preceded by a number of preprocessing steps that attempt to optimize the indexing process mainly by feature reduction, as explained in this section.

### Indexing Textual Data

Indexing is the process of mapping a document into a structured format that captures its content. It can be applied to the whole document or some parts of it, though the former is usually the case. In indexing, the terms occurring in the given collection of documents are used to represent the documents. It is widely known that text documents contain large numbers of terms that have no significant relationship to the context in which they exist. Using all the terms would certainly result in high inefficiencies; therefore, many unrelated terms are usually eliminated through some preprocessing steps, as we shall discuss later.

One very widely used indexing model is the *vector space model* (Salton & Buckley, 1988) which is based on the bag-of-words (or set-of-words) approach. This model has the advantages of relative computational efficiency and conceptual simplicity (Salton & Buckley, 1988); nonetheless, it suffers from the loss of important information about the original text, such as information on the order of the terms in the text or about the boundaries between sentences or paragraphs. In this model, each document is represented as a vector, the dimensions of which are the terms in the initial document collection. The set of terms used as dimensions is referred to collectively as the *term space*. Each vector coordinate is a term having a numeric value representing its relevance to the corresponding document with higher values implying higher relevance. The process of giving numeric

values to vector coordinates is referred to as *weighting*. From an indexing point of view, weighting is the process of giving more emphasis to more important terms.

Three popular weighting schemes have been thoroughly studied in the literature: *binary*, *term frequency (TF)*, and *term frequency by* ***inverse document frequency*** (*TF*IDF)*. For a term *t* in document *d*, the binary scheme records binary coordinate values, where a 1 is given to *t* if it occurs at least once in *d*, and a 0 is given otherwise. The TF scheme records *t*'s frequency of occurrence in *d*. It is common to normalize TF measurements in order to help overcome problems associated with document sizes. Normalization may be achieved by dividing all coordinate measurements for every document by the highest coordinate measure for that document. The TF*IDF scheme simply weights TF measurements with a global weight, the IDF (inverse document frequency) measurement. The IDF measure for a term *t* is defined as $log_2 (N/N_t)$, where *N* is the total number of documents in the collection, and $N_t$ is the total number of documents containing at least one occurrence of *t*. The reader should note that IDF increases as $N_t$ decreases, that is, as the uniqueness of the term among the documents in the given collection increases. As with TF, normalization is usually done here too. To normalize measurements based on the TF*IDF scheme, the cosine normalization is usually utilized as shown below:

$$W_{tk,dj} = \frac{TF * IDF(tk,dj)}{\sqrt{\sum_{s=1}^{|T|}(TF * IDF(ts,dj))(TF * IDF(ts,dj))}},$$

where *tk* and *dj* are the term and document under consideration, respectively, TF*IDF (*ts*,*dj*) is the coordinate measure of *ts* in *dj*, and |T| is total number of terms in term space.

## Preprocessing

Various preprocessing steps are usually performed on the text corpus prior to indexing in order to optimize the indexing process primarily by reducing the number of terms used, thus leading to faster processing at the application level later on.

***Case folding*** is the process of converting all the characters in a document into the same ***case***, either all upper ***case*** or lower ***case***. This step has the advantage of speeding up comparisons during the indexing process. ***Stemming*** is the process of removing prefixes and suffixes from words to reduce them to ***stem***s, thus eliminating tag-of-speech and other verbal or plural inflections. For example, the words "Computing," "Computer," and "Computational" all map to "Compute." It is worth noting that stemming algorithms have the disadvantage of requiring a great deal of linguistics and are, thus, language dependent. ***Stop words*** are words having no significant semantic relation to the context in which they exist. Stop words can be terms that occur frequently in most of the documents in a given collection (i.e., have low uniqueness and thus low IDF measurements) and as a result, must not be included as indexing terms. For example, articles and propositions such as "the," "on," and "with" are usually stop words. Stop words may also be document-collection specific. For example, the word "blood" would probably be a stop word in a collection of articles addressing blood infections, but certainly not in a collection describing the events of the 2006 FIFA World Cup that took place Germany.

# TEXT-DATA ANALYSIS

Text-data analysis (a.k.a. text mining) is a very broad process that can be refined into a number of task-oriented subprocesses. A treatment of the major **text-data analysis** subprocesses follows.

## Information Extraction

Many regard *information extraction (IE)* (Cowie & Lehnert, 1996) as the central text-data analysis subprocess largely owing to the success of its applications. IE has emerged as a joint research area between text-data analysis and natural language processing (NLP). It is the process of extracting predefined information on known entities and relationships among those entities from streams of documents and usually storing this information in predesigned templates. Information extraction is associated with streams of documents rather than static collections. One popular application of IE is the extraction of promotions and sales from streams of newspaper documents; the extracted information might be the event, the companies involved, or the event dates. Systems employing such technologies are usually referred to as news-skimming systems.

The IE process is twofold; first, it divides every document into relevant and irrelevant portions, and then, fills the predefined templates with the information extracted from the relevant portions. Simple IE applications, such as extracting proper names or companies from text, is currently being performed with very high precision; however, this is still not the case for more complex tasks, like determining the sequences of events from a document. In such complex tasks, IE systems are usually defined and applied on very restricted domains, normally with the help of domain experts which obviously hinders their portability to other domains. To summarize, IE systems scan streams of documents in order to transform the associated documents into much smaller bits of extracted relevant information that can be more easily maintained and comprehended. A number of very popular IE applications are briefly outlined as follows.