Text engineering and Mining

Wouldn't it be wonderful to harness available computer and language technologies to learn without reading-Alan Benson.

Section 7.0 Introduction

Heretofore data mining and analysis was limited to numeric data alone. Advances in text processing, application of statistical algorithms, natural language technologies related to semantics and concurrently developments in high speed computing have been enablers in the successful application mathematical methods to unstructured text. Large corporations, medical institutions, call centers, and disparate other entities routinely collect information in textual form. Sifting vast volumes of unstructured written text is a labor intensive and prohibitively expensive endeavor. A mechanism to automate read textual content in documents and segment into different categories is a fine proposition. This would help identify categories with similar content, which may consequently be used for business improvements, isolate problem areas etc. The literature is already replete with resounding success stories all across the board.

The moniker *text mining* came to be associated with analysis of textual data because it tends to be large. This is due to the fact that typical text contains thousands of words and other characters, numbers, addresses, and punctuation. It is inherently a complex task. Mathematical treatment of text at a semantic level to understand its message is a desirable but an unrealistic goal. Mere Words and sentences in the text do not capture linguistic subtleties, nuances, and the knowledge, perspectives, and experiences of the writer. Since it is difficult to glean semantic information from documents, except in some very specific domains, most text mining applications focus on word associations in documents. In other words, learn as much as possible by studying the co-occurrence of words in

document collections.

Section 7.1 Text mining fundamentals

The three fundamental constructs in the study of text mining are a *collection*, *document*, and a *term*. A collection is merely a corpus of documents. Documents may be variously: a paragraph, phrase, query, code fragment, e-mail, or indeed a document! A term may be a word, set of words, punctuation, or a number. A term is often referred to as a token. In practice *tokenization* or entitization of text units will lock down physical quantification of a term.

Earlier we noted that most text mining is based on extracting co-occurring word

Notes:

patterns in a document collection. It is popularly alluded to as the *bag-of words* method. In this approach the document collection or corpus is represented as a two-dimensional array of documents and terms, known as the term by document matrix. It is also known as a vector space representation of text. Given a collection of *n* documents, the corpus is represented as a matrix of order nxk. Usually the terms are rows and documents are columns. The constituent elements of the matrix are term frequencies. The term by document matrices tend to be very sparse. They contain a lot of zeroes because it is common that many terms are not pervasive across all documents. Treatment of textual data prior to analysis requires care and attention to yield meaningful results. By nature the number of words in a document is large, and a surfeit of prepositions, conjunctions, punctuation, and articles that do not contribute to distinguishing documents litter the corpus. Additionally there is typically no paucity of other non-informative terms. They contribute to escalating noise in the data and it cannot be emphasized more that care is exercised to remove them.

Section 7.2 Pre-processing Text Data

Text mining is mathematical analysis of data in large dimensions where each term is a variable. Text analysis, wherein ratio of the number of variables to total number of documents is large tends to render computation infeasible. Needless to say "*term-wise minimalism*" is the credo of the text miner. Therefore, significant pre-processing is done prior to the creation of the term by document matrix. A sequence of preliminary steps is undertaken prior to analysis.

• Stop and synonym lists

First non-value added terms are discarded using *stop lists*. Stop lists are user created library of terms to be excluded from analysis. The number of terms is further reduced, using *synonym lists*. A synonym list is a pre-determined set of word pairings establishing term equivalence. In other words, the equivalence (e.g., laptop = notebook) bundles laptops and notebooks together and establishes a single representative term "notebook." Creating synonym lists requires support from different languages as many words have synonymous foreign cousins.

• Stemming words in a document

A procedure called *stemming* is applied to map multiple terms to a single "parent" root term. Stemming is a morphological operation that reduces variants of a term to their form, i.e., multiple words are mapped to a single root term (e.g., {working,

Notes:

worked, worker,..}=work). Some text mining tools provide another facility called *start lists*. A start list merely is a user created word list. Only terms in the start list are admitted in later processing. Creating a start list requires extensive and comprehensive domain knowledge.

•Other problems with text

• Part of speech tagging

Also, many programs offer parts-of-speech tagging, and conflating nouns to form phrases. Conflation allows treating the same word as different words based on context. While, these techniques are valuable to derive semantic meaning, they augment the number of terms in the corpus. Clearly there is a trade-off between number of dimensions and using variables that do not cooperate in the data reduction effort. So such options may be exercised judiciously. These techniques help overcome the *curse of dimensionality* which is a miner's nightmare. In spite of noise reduction efforts to make the analysis tractable, the corpus still will contain several thousand terms and problems related to *polysemy* and *synonymy* still

Notes:

remain.

• Polysemy and synonymy

Polysemy is a condition where the meaning of a word is context dependent. The word “bank” is an example. It may refer to a river bank or a financial institution depending on the situation. This is particularly sticky because unrelated documents tend to get clustered together because the terms themselves, but not their import is used to characterize them. Synonymy relates to a condition where two terms “retire” and “sunset” in a particular parlance convey the meaning that a product or an application is discontinued. However documents with these terms tend to be unrelated although there is a latent semantic relationship between them. Yet another phenomenon prevalent in text data is known as *term dependency*. Term dependency is the tendency for words to occur together. For example consider an on-line store selling servers. In this context, the terms “server” and “blade” occur together and are strongly correlated. Mere bag-of words approach will be unable to

Notes:

capture such joint occurrences of seemingly unrelated terms.

Some text mining tools treat *capitalized* and *un-capitalized* versions of the same word as two different terms. This unnecessarily and needlessly increases the number of variables. It is advised that the text to be parsed for mining is uniformly lower-case or upper-case across all documents. The "Roll-Up Terms" method sometimes used for text mining admits only a specified number of weighted terms. Typically the top 100 highest weighted terms are included. To further exacerbate problems, textual data is replete with miss-spellings, slang, acronyms, foreign vocabulary and other gibberish. One solution is to run the "spell-checker" tool available in most commercial applications. While the effort may be tedious, the benefit is considerable. In so far as the other ailments such as slang, acronyms, and others, manual data mopping is the only recourse.

Notes:

Figure 7.1 is a schematic of the pre-processing steps and documents clustering

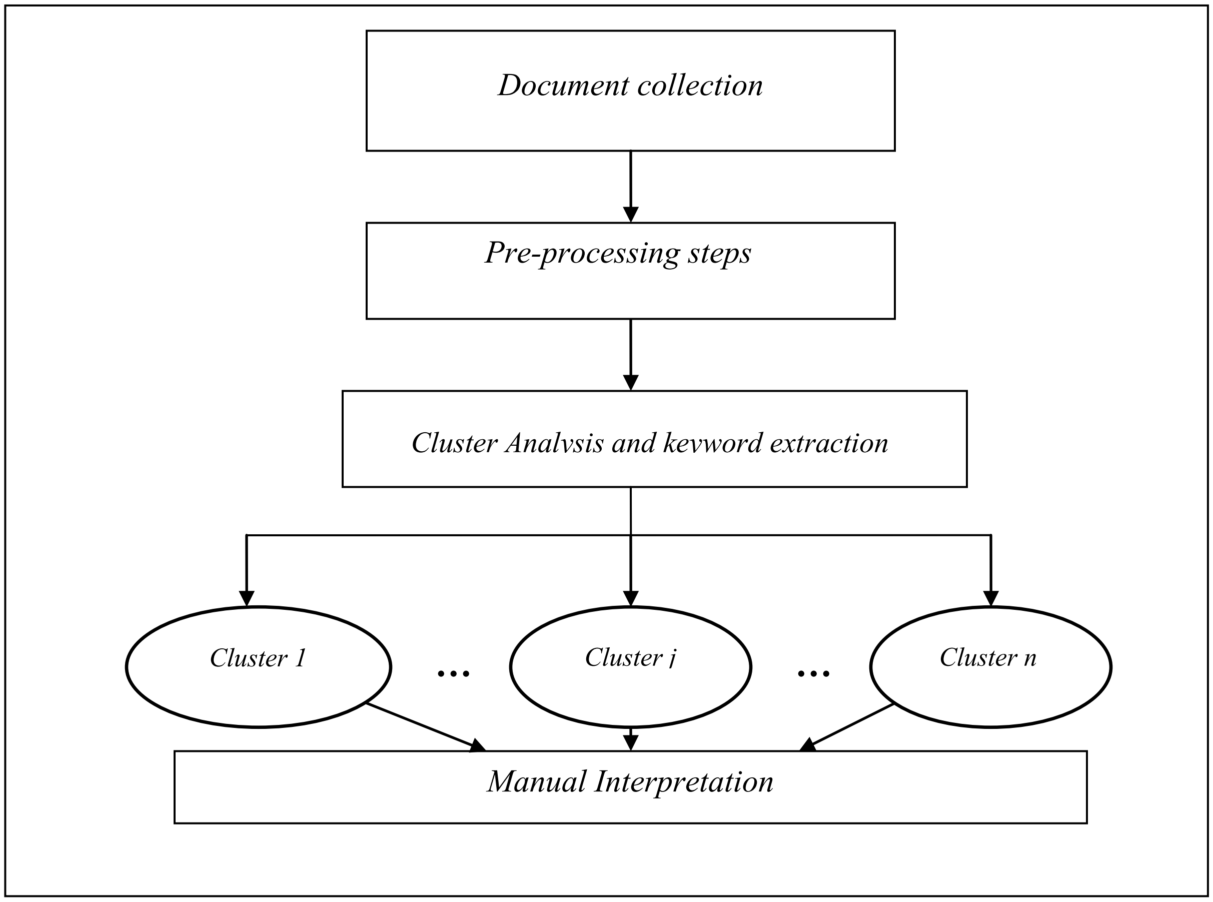

Figure 7.1. Schematic of the steps involved in text analysis

- Compile a comprehensive list of all words that appear in the collection.
- Drop articles, conjunctions, and prepositions
- Discard pronouns
- Delete non-valued added verbs (see, do, be, thank…) and adjectives (great, cool,.)
- Delete terms that occur in all documents
- Delete terms that appear in only one document.

- Foreign words, and acronyms such as i.e., e.g., etc., and words like such, thus, should be dropped.

Section 7.3 Frequency weights and inverse document frequency weights

A final pre-processing step requires that the terms are weighted suitably as not all words are treated “equal” in the world of text mining. There are two types of weights; *frequency-weights*, and *inverse document frequency weights.* The frequency-weight is given by:

$$t_f = \log_2(f_{ij}) + 1 \qquad (7.1)$$

Where f_{ij} is the frequency of the $"i^{th}"$ term in the $"j^{th}"$ document. If the $"i^{th}"$ term occurs in the $"j^{th}"$ document once, then f_{ij} is equal to one such that $\log_2(1) = 0$ yielding zero weight. Adding “one” gives a unit weight to the singleton observation. The purpose of the logarithmic transformation is to attenuate the effect of weights as well as provide a smooth weight assignment scheme. For example, merely using f_{ij} as a weight, would make it equal to 7 and 8 for those observations

Notes:

occurring 7 and 8 times respectively. Frequency weight defined by t_f gives 3.8 and 4 respectively-a more appropriately apportioned and smooth weighting! The inverse document frequency is given as

$$idf = \log_2\left(\frac{n}{d_j}\right) + 1 \qquad (7.2)$$

Where n is the document collection size and d_j is the number of documents hosting the "i^{th}" term. The weight $w_j = t_f\,idf$. In lieu of inverse document frequency, *entropy* is used, and is given by

$$E = \sum_i \frac{p_{ij}\log_2(p_{ij})}{\log_2(n)} + 1 \qquad (7.3)$$

Where $p_{ij} = \dfrac{f_{ij}}{d_j}$. The terms in the document are weighted using w_j. There are various other versions of term weights and frequency weights and a brief discussion follows. Another smoothed version of inverse document frequency is one which recognizes a condition that drives the weight to indeterminacy is:

Notes:

$$idf = \log\left(\frac{n - d_i + 0.5}{d_i + 0.5}\right) \tag{7.4}$$

Section 7.4 Singular Value Decomposition (Technical details optional)

The bag-of-words techniques do not address the laundry list of problems aired in section 7.2. A popular method called *singular value decomposition* is applied to resolve some of the problems addressed. Singular value decomposition (SVD) is a matrix decomposition technique applied to matrices. Applied to term by document matrices, it is known as latent semantic indexing (LSI). It is simply a scheme to project vectors in a high dimensional space to a lower dimension such that correlated terms that occur together are projected onto the same dimension. It may also be viewed as a dimensionality reduction technique. In the following we will give a mathematical exposition.

Notes:

Definition 7.1

Let *D* be an *m* x *n* term by document matrix whose elements are term frequencies.

The rank of *D* is *r*, $r \le n$.

The singular value decomposition factors matrix *D* into three matrices.

$$D = U\Sigma V^T \tag{7.5}$$

Where *U* is an orthogonal matrix of order *m* x *k*, Σ is a diagonal matrix of singular values of order *k* x *k* and *V* is orthogonal with order *k* x *n*. Matrix D can be expressed as a sum of *k* matrices as shown below.

$$D = U\Sigma V^T = \sum_{i=1}^{k} \lambda_i U_i V_i^T \tag{7.6}$$

The power Singular value decomposition is that it is a matrix approximation technique. Matrix D can be approximated by $D_j, j \le k$. That is the matrix D_j is the best approximation of rank *j* to D.

As an example, we will demonstrate the matrix calculations involved and the concepts via *2* x *2* matrices.

Notes:

$$D = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix}^T = \begin{bmatrix} \lambda_1 u_{11} v_{11} + \lambda_2 u_{12} v_{12} & \lambda_1 u_{11} v_{21} + \lambda_2 u_{12} v_{22} \\ \lambda_1 u_{21} v_{11} + \lambda_2 u_{22} v_{12} & \lambda_1 u_{21} v_{21} + \lambda_2 u_{22} v_{22} \end{bmatrix} \text{(7.7)}$$

Expressing D as a summation, λ_i is the i^{th} singular value, U_i is the i^{th} column of matrix *U*, and V_i^T is the i^{th} column of *V*. Thus the sum of matrices D_1 and D_2:

$$D_1 = \lambda_1 \begin{pmatrix} u_{11} \\ u_{21} \end{pmatrix} \begin{pmatrix} v_{12} & v_{21} \end{pmatrix} = \lambda_1 \begin{pmatrix} u_{11} v_{11} & u_{11} v_{21} \\ u_{21} v_{11} & u_{21} v_{21} \end{pmatrix}, \; D_2 = \lambda_1 \begin{pmatrix} u_{12} \\ u_{22} \end{pmatrix} \begin{pmatrix} v_{12} & v_{22} \end{pmatrix} = \lambda_2 \begin{pmatrix} u_{12} v_{12} & u_{12} v_{22} \\ u_{22} v_{12} & u_{22} v_{22} \end{pmatrix}$$

is given by D.

So D can be represented as a sum of matrices. Note that D_1 and D_2 are rank one. One definition of the rank of a matrix is highest order non-zero determinant.

$| D_1 | == \begin{vmatrix} u_{11} v_{11} & u_{11} v_{21} \\ u_{21} v_{11} & u_{21} v_{21} \end{vmatrix} = u_{11} v_{11} u_{21} v_{21} - u_{21} v_{11} u_{11} v_{21} = 0$. Thus the highest order non-zero determinant is 1.

One can get a rough approximation of D by computing D_1. Of all the possible matrices, $\arg\min_i \| D - D_i \|, i = 1,2.....k$ is the best approximation, when $D_i = D_1$ where ||.|| is the L^2-norm. The approximation may be improved by adding D_2 to D_1.

Notes:

The matrix $D_1 + D_2$ is the best approximation of the original matrix D of rank two. For higher order matrices, successive additions of rank 1, 2, 3, ,... *k* yield better approximations to D.

There is a close relationship between singular value decomposition of a matrix D and *spectral decomposition*. Spectral decomposition is simply the matrix of Eigen values of $D^T D$. In the following we will discuss the mathematical relationship between SVD and spectral decomposition. Spectral decomposition of a matrix $\underset{\sim}{X}$ of order *pxp* is simply $\underset{\sim}{X} = P^T \Lambda P$, where P is an orthogonal matrix of order pxp whose columns are Eigen vectors. The matrix Λ is a diagonal matrix of order pxp whose on-diagonal elements are eigen values. Note that spectral decomposition is possible when the $\underset{\sim}{X}$ is symmetric and positive definite (A matrix is said to be positive definite when its Eigen values are all positive (see appendix). When the matrix $\underset{\sim}{X} = D^T D$ is formed from a non-square matrix, the singular values

Notes:

of D are simply the square roots of the eigen values of matrix $\underset{\sim}{X}$. Furthermore, the Eigen vectors of $D^T D$ are the right hand singular vectors of D, and the Eigen vectors of DD^T are the left hand singular vectors of D. This assertion is quite easy to show.

$$D^T D = V\Sigma U^T U\Sigma V^T = V\Sigma I\Sigma V^T = V\Sigma^2 V^T \qquad (7.8)$$

This is a handy result, in that one way to compute the singular values of a matrix D is to form the product matrix $D^T D$ and compute its Eigen values and corresponding Eigen vectors. As stated earlier, the square roots of the Eigen values of product matrix $D^T D$ give the singular values of D and the Eigen vectors are the right singular vectors. Similarly $D^T D$ gives the left singular vectors.

We pointed at the beginning of the section that a *bag-of words* approach to analysis of textual data does not comprehend *textual nearness*. The measurement of textual nearness is more from a semantic sense than spatial, although spatial nearness is intrinsic in the text structure. As an example consider the words "misclassification"

Notes:

and “rate.” These seemingly unrelated terms occur always together in the context of a classifier performance. Singular value decomposition helps organizing terms in lower dimensions than the original larger dimensional term hyperspace. Using the matrix decomposition studied earlier in the section, the documents can be projected into a lower k -dimensional space. The matrices $U_{(k)}$ and $V_{(k)}$ form an orthonormal basis (see Appendix) for the k dimensional document and term spaces respectively. This terrifyingly arcane mathematical language simply means that the columns of these two matrices form a new set of axes obtained as a result of the rotation of original axes. Therefore, a document d_i strung out as a *p*-dimensional vector of terms is projected onto the lower k -dimensional space such that

$d_i^{'} = U_{(k)}^T d_i$. Thus, $d_i^{'}$ is a k -dimensional vector whose $"j^{th}"$ element is a sum formed by multiplying elements in the j^{th} singular vector and the original term frequencies in the document. The coordinates of the j^{th} singular vector may be viewed as weights applied to terms in the corpus. The text miner interface in SAS allows the user to

Notes:

choose the SVD technique. It however requires the user to input the number of SVD variables and any selection may be arbitrary. One way to interpret singular values is that they capture the total amount of variation in the document collection. Thus, the more singular values we incorporate into the analysis, the better. But, due to issues of curse of dimensionality, one would like to be frugal. Since text miner does not provide a nice way to capture total variation, it is not possible to determine proportion of variation explained by the number of singular value dimensions chosen. SAS recommends that depending on the collection, anything in the range of 10-250 ought to perform well. This recommendation is based purely on anecdotal experience, and there is no telling if it will work on some datasets.

Example 7.1

The following is an example of a document collection obtained as a result of delivering on-line survey forms to visitors at a large commercial website. A portion of the survey form is reserved for the visitor to write about their grievances and other

Notes:

comments. Thus each completed survey form represents a document and the entire collection survey forms; the document corpus. After thorough pre-processing involving the steps delineated previously, clustering based on SVD inputs produced keyword outputs by clusters. A month by month analysis is depicted in Table Ten. A Gaussian mixture model approach was adopted to perform clustering analysis. Within Enterprise Miner in SAS, the Gaussian mixture model is called the E-M method which is the expectation maximization algorithm introduced in a previous chapter.

Cluster description	June	July	August	September	October
Printer supplies related problems	color, print, printer, hp, accessory (7%)	cartridge, color, print, printer, hp (7%)	cartridge, color, printer, print, model (7%)	printer, cartridge, print, list, hp (7%)	color, print, printer, item, give (4%)
Part Number and search problems	engine, search, specific , result, (4%)	numb, part number, part, compaq, model (4%)	numb, part number, part, order, compaq (5%)	numb, part number, part, spare, compaq (5%)	numb, part, model, find, specifications (4%)
Driver download problems	download, driver, patch, update, software (6%)	download, driver, web site, web, hp (18%)	download, driver, update, software (6%)	download, link, driver, page, appear (5%)	download, driver, software, unix, update (9%)
Server configuration problems	server, configure, option, model, technique (5%)	server, configure, buy, hard, find (6%)	server, info, storage, customize, configure, (5%)	server, web site, web, software, unix (14%)	product, server, price, detail, information (12%)
Page rendering performance problems	load, page, slow, time, long (10%)	load, page, slow, time (9%)	load, page, slow, time, long (8%)	load, page, slow, time, appear (9%)	load, long, page, slow, time (10%)
PocketPC upgrade problems	link, pda, purchase, upgrade, order (6%)	link, pda, purchase, upgrade, software (6%)			
Subscription login problems					address, change, email, log, login (14%)

Table 7.1. Results of clustering textual data over time including keywords and percentage of total

The interesting finding is that clustering is able to find “PocketPC upgrade problems” that occurred in August and persisted through September. The problem resulted due to a link embedded in an e-mail, sent to pocket PC owners failed to render the page with upgrade information. Being able to uncover such valuable information is the strength of clustering.

References

- Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, Ramasamy Uthurusamy, Advances in Knowledge Discovery and Data Mining, MIT press, 1996.
- Berthold, Michael, Hand, J., David, Intelligent data analysis, Springer, 1999
- Hastie Trevor, Tibshirani, Robert, Friedman, Jerome, Elements of statistical learning, Data mining, Inference, and Prediction, Springer, 2001.
- Duda, R., Hart, P., and Stork, D. Pattern Classification, 2nd edition, John Wiley Interscience, 2001.
- Jiawei Han and Micheline Kamber, Data Mining: Concepts and Techniques, The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor Morgan Kaufmann Publishers, August 2000
- SAS Language Reference version 6", SAS Institute (1993) ISBN 1-55544-381-8.
- SAS/STAT User's Guide, volume 1" version 6 fourth edition, SAS Institute (1990) ISBN 1-55544-376-1.
- SAS/STAT User's Guide, volume 2," version 6 fourth edition, SAS Institute (1990) ISBN 1-55544-376-1.
- SAS Procedures Guide," version 6 third edition, SAS Institute (1990) ISBN 1-55544-378-8.
- Pyle, Dorian, Data Preparation for Data Mining, Morgan Kaufmann Publishers, 1999
- Hand D.J., Mannila H., and Smyth P. (2001) *Principles of data mining*, MIT Press.
- R.A. Johnson, and D.W. Wichern, Applied Multivariate Statistical Analysis, 3rd edition, Englewood Cliffs, New Jersey: Prentice Hall, 1992
- R. O. Duda, and P.E. Hart, and Stork, David, G. Pattern Classification and Scene Analysis, New York: John Wiley Interscience, 2001.
- Letsche, T.A. and Berry M.W., Large scale information retrieval with latent semantic indexing, Information sciences, 100, 105-137, 1997
- Berger, James, A, Statistical Decision Theory and Bayesian Analysis, 2nd edition, Springer-Verlag, 1985
- Schapire, R.E., and Freund, Y. 2000Boostexter: a boosting based system for text categorization. Machine Learning 39, 135-168. 2000.
- Baldi, P., Frasconi, P., and Smyth, P. Modeling the Internet and the Web. Probabilistic Methods and Algorithms, John Wiley, 2003.
- Mitchell, T., Machine Leaning, John Wiley, 1997.
- Effron, B. and Tibshirani, R. An introduction to the Bootstrap, Chapman and Hall, 1993
- Sholom Weiss, Nitin Indurkhya, Frederick Damerau, Text Mining: Methods for analyzing unstructured information
- Kutner, Michael, Nachtsheim, John Neter, William Li, Applied Linear Statistical Models, 5 edition, McGraw-Hill/Irwin, 2004