AspectAnalyzer - distibuted system for bi-clustering analysis

Pawel Foszner

2015

Contents

List of Figures

List of Tables

1 Introduction

2 Installation

2.1 System Requirements

For proper working of AspectAnalyzer software, the following requirements should be met:

- Windows operating system 7 or later
- .NET Framework 4.0 or later
- 40MB of free space for executables
- 100MB or more of free space for stored results
- MSMQ Queue feature enabled in the system
- Account with administrative privileges

2.2 Installing Message Queuing (MSMQ)

To install Message Queuing 4.0 on Windows Server 2008 or Windows Server 2008 R2:

1. In Server Manager, click Features.
2. In the right-hand pane under Features Summary, click Add Features.
3. In the resulting window, expand Message Queuing.
4. Expand Message Queuing Services.
5. Click Directory Services Integration (for computers joined to a Domain), then click HTTP Support.
6. Click Next,then click Install.

To install Message Queuing 4.0 on Windows 7 or 8 or Windows Vista:

1. Open Control Panel.
2. Click Programs and then, under Programs and Features, click Turn Windows Features on and off.
3. Expand Microsoft Message Queue (MSMQ) Server, expand Microsoft Message Queue (MSMQ) Server Core, and then select the check boxes for the following Message Queuing features to install:
   (a) MSMQ Active Directory Domain Services Integration (for computers joined to a Domain).
   (b) MSMQ HTTP Support.
4. Click OK.

5. If you are prompted to restart the computer, click OK to complete the installation.

To install Message Queuing 3.0 on Windows XP and Windows Server 2003

1. Open Control Panel.
2. Click Add Remove Programs and then click Add Windows Components.
3. Select Message Queuing and click Details.

Note:
If you are running Windows Server 2003, select Application Server to access Message Queuing

4. Ensure that the option MSMQ HTTP Support is selected on the details page.
5. Click OK to exit the details page, and then click Next. Complete the installation.
6. If you are prompted to restart the computer, click OK to complete the installation.

2.3 Checking version of .NET Framework installed

To find .NET Framework versions by viewing the registry (.NET Framework 1-4)

1. On the Start menu, choose Run.
2. In the Open box, enter regedit.exe. You must have administrative credentials to run regedit.exe.
3. In the Registry Editor, open the following subkey:

   `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP`

   The installed versions are listed under the NDP subkey. The version number is stored in the Version entry. For the .NET Framework 4 the Version entry is under the Client or Full subkey (under NDP), or under both subkeys.

To find .NET Framework versions by viewing the registry (.NET Framework 4.5 and later)

1. On the Start menu, choose Run.
2. In the Open box, enter regedit.exe. You must have administrative credentials to run regedit.exe.
3. In the Registry Editor, open the following subkey:

   `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full`

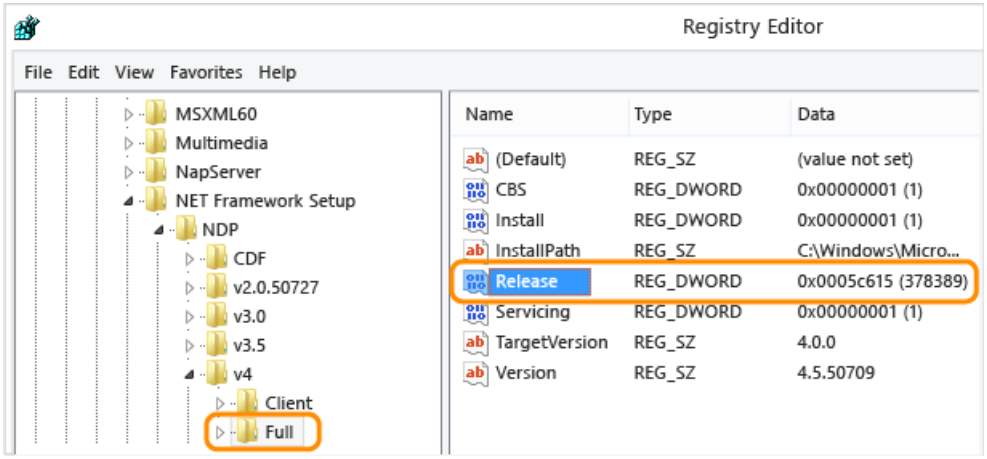

Figure 1: .NET Framework Release number

Check for a DWORD value named Release. The existence of the Release DWORD indicates that the .NET Framework 4.5 or newer has been installed on that computer.

Value of the Release DWORD	Version
378389	NET Framework 4.5
378675	NET Framework 4.5.1 installed with Windows 8.1
378758	NET Framework 4.5.1 installed on Windows 8, Windows 7 SP1, or Windows Vista SP2
379893	.NET Framework 4.5.2

Table 1: .NET Framework Release numbers

2.4 Installation Wizard

3 Input data

4 Algorithms

4.1 Algorithms based on matrix decomposition

A very wide range of algorithms are algorithms based on data matrix decomposition. In such methods data matrix (A) is factorized into (usually) much smaller matrices. Such a distribution, because of the much smaller matrices is much easier to analyze, and the obtained matrices reveal previously hidden features. These algorithms are often called NMF algorithms. NMF stands for non-negative matrix factorization. Two efficient algorithms were introduced by

Seung and Lee [8]. First minimize conventional least square error distance function and second generalized Kullback–Leibler divergence. Third and last from this group is algorithm that slightly modify the second approach. Author [22] introduce smoothing matrix for achieving a high degree of sparseness, and better interpretability of the results. Data matrix in this techniques is factorized into (usually) two smaller matrices:

$$A \approx WH \tag{1}$$

Finding the exact solution is computationally very difficult task. Instead, the existing solutions focus on finding local extrema of the function describing the fit of the model to the data. Below some examples of such divergence functions.

4.1.1 Based on LSE

.

Distance function:

$$\|A - WH\|^2 = \sum_{ij}(A_{ij} - WH_{ij})^2 \tag{2}$$

Update rules:

$$H_{ij} = H_{ij}\frac{(W^TV)_{ij}}{(W^TWH)_{ij}} \tag{3}$$

$$W_{ij} = W_{ij}\frac{(VH^T)_{ij}}{(WHH^T)_{ij}} \tag{4}$$

4.1.2 Based on Kullback–Leibler divergence

Divergence function:

$$D(A\,||\,WH) = \sum_{ij}(A_{ij}log\frac{A_{ij}}{WH_{ij}} - A_{ij} + WH_{ij}) \tag{5}$$

Update rules:

$$H_{ij} = H_{ij}\frac{\sum_k W_{ki}V_{kj}/(WH)_{kj}}{\sum_l W_{li}} \tag{6}$$

$$W_{ij} = W_{ij}\frac{\sum_k H_{jk}V_{ik}/(WH)_{ik}}{\sum_l H_{jl}} \tag{7}$$

4.1.3 Based on non-smooth Kullback–Leibler divergence

Divergence function:

$$D(A \,||\, WSH) = \sum_{ij}(A_{ij} log \frac{A_{ij}}{WSH_{ij}} - A_{ij} + WSH_{ij}) \tag{8}$$

Update rules for this method is the same as in previews one, but instead W in update rule for H we substitute WS, and in update rule for W we substitute SH. Smoothing matrix S looks as follows:

$$S = (1 - \theta)\mathbf{I} + \frac{\theta}{q}\mathbf{1}\mathbf{1}^T \tag{9}$$

Where: I – Identity matrix, 1 – vector of ones and Θ – should meet condition $0 \leq \Theta \geq 1$. Another type of group NMF algorithms are algorithms based on the expectation-maximization method. Because of the approach, the distance function replaces the likelihood function. Below the examples of such methods.

4.1.4 PLSA

PLSA stands for Probabilistic Latent Semantic Analysis. Introduced by Thomas Hoffman [1], and based on maximizing log-likelihood function. For this purpose author use Expectation-Maximization algorithm [5]. Formulas for computing results:

Log-likelihood function:

$$E\left[L^C\right] = \sum_{i=1}^{N}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} P(z_k \mid d_i, w_j log\left[P(w_j \mid z_k)P(z_k \mid d_i)\right] \tag{10}$$

E-step:

$$P(z_k \mid d_i, w_j) = \frac{P(w_j, z_k)P(z_k, d_i)}{\sum_{l=1}^{K} P(w_j, z_l)P(z_l, d_i)} \tag{11}$$

M-step:

$$P(w_j \mid z_k) = \frac{\sum_{i=1}^{N} n(d_i, w_j)P(z_k, d_i, w_j)}{\sum_{m=1}^{M}\sum_{i=1}^{N} n(d_i, w_m)P(z_k, d_i, w_m)} \tag{12}$$

$$P(z_k \mid d_i) = \frac{\sum_{i=1}^{N} n(d_i, w_j)P(z_k, d_i, w_j)}{n(d_i)} \tag{13}$$

The author explains the meaning of those formulas by using the example. Factor w_j represent one word from vocabulary that contains M words. Factor d_i represents one of N documents. And z_k means aspect. Expression $n(d_i)$ denotes number of words in document i, and $n(d_i, w_j)$ denotes number of occurrences of word j in document i.

Translating the data generation process into a joint probability model results in the expression:

$$P(w_j \mid d_i) = \sum_{k=1}^{K} P(w_j \mid z_k) P(z_k \mid d_i) \tag{14}$$

In above equation all possible probabilities $P(w_j \mid d_i)$ form a data matrix (in our notation V) with M rows and N columns. Authors assume that this matrix contains K bi-clusters. Data matrix is factorized into two smaller matrices. The first one has M rows and K columns, and represents the probability of occurrence of a word in the context of aspect. The second consists of K rows and N columns, and represents probability of an aspect in the document. Single bi-cluster is in the matrix formed from the product of k-th column from first matrix and k-th row.

4.1.5 FABIA

FABIA stands for Factor Analysis for BIClustering Acquisition. Algorithm were introduced by Hochreiter [23] and based on Expectation-Maximization algorithm.

E-step

$$E(z_j \mid x_j) = (\Lambda^T \Psi^{-1} \Lambda + \Xi_j^{-1})^{-1} \Lambda^T \Psi^{-1} x_j \tag{15}$$

$$E(z_j z_j^T \mid x_j) = (\Lambda^T \Psi^{-1} \Lambda + \Xi_j^{-1})^{-1} + E(z_j \mid x_j) E(z_j \mid x_j) \tag{16}$$

Where Ξ_j stands for $diag(\xi_j)$, where update for ξ_j is:

$$\xi_j = diag(\sqrt{E(z_j z_j^T \mid x_j)}) \tag{17}$$

M-step:

$$\Lambda^{new} = \frac{\frac{1}{l} \sum_{j=1}^{l} x_j E(z_j \mid x_j)^T - \frac{\alpha}{l} \Psi sign(\Lambda)}{\frac{1}{l} \sum_{j=1}^{l} E(z_j z_j^T \mid x_j)} \tag{18}$$

$$diag(\Psi^{new}) = diag(\frac{1}{l} \sum_{j=1}^{l} x_j x_j^T - \Lambda^{new} \frac{1}{l} \sum_{j=1}^{l} E(z_j \mid x_j) x_j^T) + diag(\frac{\alpha}{l} \Psi sign(\Lambda) (\Lambda^{new})^T) \tag{19}$$

Where:

- z – vector of factors,
- x – sample from data matrix,
- Λ – sparse prototype matrix,
- Ψ - covariance matrix – expressing independent noise,
- ξ – variational parameter,
- l – number of factors.

Data initialization:

1. vectors ξ_j by ones
2. Λ randomly
3. $\Psi = diag(max(\delta, covar(x) - \Lambda\Lambda^T))$

Model likelihood is define as follows:

$$p(x \mid \Lambda, \Psi) = \int p(x \mid z, \Lambda, \Psi)p(z)dz \tag{20}$$

Where:

$$p(z) = \left(\frac{1}{\sqrt{2}}\right)^2 \prod_{i=1}^{p} e^{-\sqrt{2}|z_i|} \tag{21}$$

Likelihood function introduce a model family that is parameterized by ξ, where the maximum over models in this family is the true likelihood:

$$\underset{\xi}{argmax} p(x \mid \xi) = p(x) \tag{22}$$

4.2 Algorithms based on bipartite graphs

4.2.1 QUBIC

QUBIC stands for QUalitative BIClustering algorithm. It was proposed by Guojun Li, et al. [5] as very efficient algorithm for analysis of gene expression data. Authors proposed weighted graph representation of discretized expression data. The expression levels are discretized to the ranks. Their number is determined by the user through the parameters of the algorithm. Number of ranks is essential and strongly affects the results. The algorithm allows two types of ranks. The positive (for up-regulating genes) and negative sign (for down-regulating genes). The vertices of the graph represent genes. The edges between them have weight to reflect the number of conditions for which they have the same rank.

Algorithm starts with translating data matrix into new representation, which is a graph where vertex set is built from rows. An intermediate step is to create a matrix of integers. This matrix is the same size as original data matrix and its values are created as follows:

1. For each row i all values are sorted in increasing order:

$$a_{i,1} \dots a_{i,s-1} a_{i,s} \dots a_{i,c-1} a_{i,c} a_{i,c+1} \dots a_{i,m-s+1} a_{i,m-s+2} \dots a_{i,m} \tag{23}$$

Where:

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array}\begin{bmatrix} -1 & 2 & -2 & 0 & 1 \\ 1 & 0 & -2 & 2 & -1 \\ -1 & 2 & 0 & -2 & 1 \\ 1 & -1 & 0 & 2 & -2 \\ 2 & -1 & 1 & 0 & -2 \end{bmatrix}$$

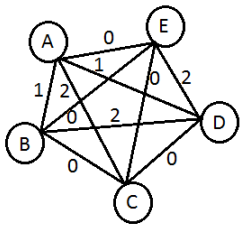

Figure 2: Sample QUBIC transformation from matrix of integers to final graph

m – number of columns $c = \frac{m}{2}$ – the median value in a row $s = m * q + 1$ – number which determine how many values will be marked as zero. q is parameter selected by the user

2. Values are marked as zero if a_{ij} belongs to interval $(a_{ic} - d_i, a_{ic} + d_i)$ where $d_i = min(a_{ic} - a_{is}, a_{i,m-s+1} - a_{ic})$
3. Values are marked with positive ranks from range $< 1, r >$ if $a_{ij} > a_{ic} + d_i$
4. Values are marked with positive ranks from range $< 1, r >$ if $a_{ij} < a_{ic} - d_i$

Figure 9. .

Bi-clusters are find one-by-one. Starting from single heaviest and unused edge as seed, algorithm iteratively add additional edges until its violates pre-specified consistency level.

5 Node manager

AspectAnalyzer software is able to create network for distributed computations. For this purpose, on all computers that will serve as network nodes, user should install the AspectAnalyzer software. For the network to work properly there should be one master node and at least one slave nodes (the more the better). For each instance of slave, user should set the IP address of the master in its configuration. Master node will configure automatically when the slave nodes start report. Example Network with four slave nodes is shown on figure 3.

Slaves nodes report about their status every 5s (how many and which jobs are in the queue). These are very short messages with information about the current load. This is handled by the MSMQ queues, and does not significantly affect the system performance. The master node allocate new tasks based on (1) the quantity of jobs in the slaves queue, and (2) the size of these tasks (data matrix size). The whole process, user can keep track via the "Node Manager" panel. In addition to the informational value of this panel, it also allows to exclude the node from the network and/or stop the tasks that are being performed on it.

Configuration of distributed computing is possible only with the external database (Microsoft SQL Express 2008 or grater). The results obtained by slave nodes

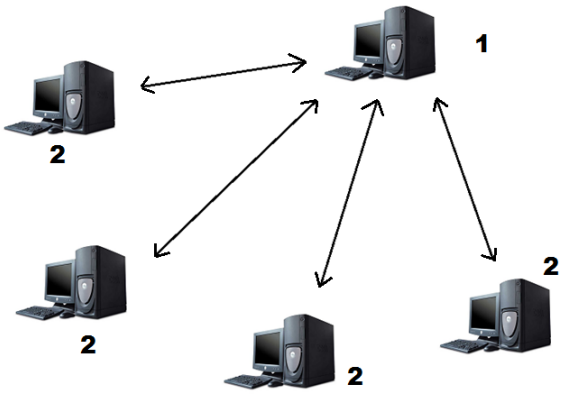

Figure 3: (1) Master - (2) Slaves network created by AspectAnalyzer

are entered by them directly to the base (the address of which is given in the configuration of each).

6 Result Viewer

7 Software Update

8 About