1 Product Introduction

ChiST Algorithm is an algorithm designed for the task of extracting the correlations among the words in a document collection. The algorithm employs a point-wise approximation in accordance with the Chi-squared distribution on the correlation among the emerging words, based on lexical collocation information.

1.1 Algorithm

Formula 1 shows the statistical method of the ChiST Algorithm. Each of the expected value and the observed value are compared and the null hypothesis is rejected when the difference, which follows the Chi-squared distribution, reaches the threshold. The Chi-square is calculated in accordance with the Corrected Pearson's Chi-squared Test of Homogeneity. [4] This calculation method for the Chi-square is corrected by the Yates' continuity correction, which is based on the asymptotic assumtion that the Chi-square test suffers from the problem of distribution estimation.

$$\chi^2_{h,cor} = \frac{N(|O_{11}O_{22} - O_{12}O_{21}| - N/2)^2}{R_1R_2C_1C_2} \tag{1}$$

Formula 2 displays how the Chi-square values are normalized for the use within a set scale of [0,1]. Most values, however, converge on 0 at this stage.

$$\chi^2_{norm[0,1]} = \frac{\chi^2 - min([\chi^2]^T)}{max([\chi^2]^T) - min([\chi^2]^T)} \tag{2}$$

Formula 3 further transforms the Chi-squared distribution into a machine-learnable feature vectors by applying [3], which is known to robustly explain the similarity between two data and reflect the symmetric property of the matrix, to modify the correlation function in Formula 1.

$$\lambda(X_1, X_2) = e^{-\gamma(1-corr(X_1,X_2))} \tag{3}$$

1.2 Characteristics

The figure below is are Chi-square distribution graphs produced by Formula 2 and 3. It shows how the Chi-square distribution is normalized by Formula 2 and how it is transformed by Formula 3. Such process allows for the acquisition of the values for building a space appropriate for machine-learning tasks.

1.3 Time Complexity

ChiST Algorithm has a time complexity of $\frac{\alpha\beta O(n^2)}{\gamma\zeta}$. Parallel processing is conducted through the Collapsed Matrix Method. (alpha:the size of the stopwords, beta:, gamma:number of CPU cores, zeta:resistance value)

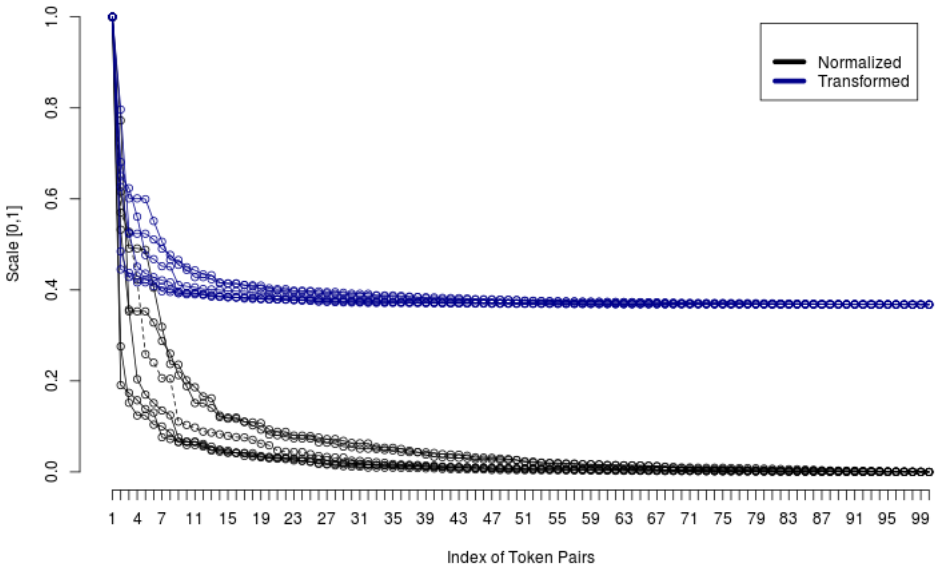

Figure 1: Normalized and transformed Chi-square distributions of token pairs [2]

2 Functional Specification

2.1 Dependencies

The following depedencies are required for the installation of the ChiST Algorithm in a Linux (RHEL6/CentOS6/Ubuntu/Debian) 32/64bit environment: gcc version 4.6.3, Cython version 0.15.1. The installation of OWLNEST nlpstat mathematical statistics library is also required. (provided upon installation)

2.2 Methods

- remove_html_tags(String data)
- remove_url(String data)
- remove_email(String data)
- RunPartitioner(File InputFile, int PartNum)
- PartitionerByCoreForBigData(File InputFile, int PartNum)
- sayTurnaroundTime(TIME sTime, TIME eTime)
- removeStopPattern(String data, List StopPattern)
- removeStopWords(String data, List stopwords)
- cleanTextForEnglish(String data)
- cleanTextForChinese(String data)

- cleanTextForKorean(String data)
- fileAggregator(Time currentTime)
- calcMultiChiSquareForOneWORD(String WORD, int uwi, String uwkey, Hash UniqWordsDictOverThreshold, Hash TokenIndexTable, int SPAN_IDX, int TotalCnt, int ShowMeMode, int LHFLag)
- calcChiSquareModuleForOneWORD(String WORD, Hash UniqWordsDictOverThreshold, Hash TokenIndexTable, int SPAN_IDX, int TotalCnt, int ShowMeMode, int LHFlag, TIME currentTime)
- calcMultiChiSquare(String WORD, int uwi, String uwkey, Hash UniqWordsDictOverThreshold, Hash TokenIndexTable, int SPAN_IDX, int TotalCnt, int ShowMeMode, int LHFLag)
- calcChiSquareModulD(String WORD, Hash UniqWordsDictOverThreshold, Hash TokenIndexTable, int SPAN_IDX, int TotalCnt, int ShowMeMode, int LHFlag, TIME currentTime)
- getSpanChiSquareStat(FILE InputFile, int SPAN_IDX, String StopWords, List StopPattern, int ShowMeMode, int KeyMode, WORD, int RunFlag, int LHFlag)
- getPairLikelihood(char* W1, char* W2, int vor, int nach)

3 Technical Support

3.1 Policy

ChiST Algorithm must be installed by a technician from OWLNEST on site/via remote. Policies on the source code is only included in the Fully Unlimited license.

3.2 Contact

- Person-in-Charge : Sunghee Kang
- Email : contact@owl-nest.com
- Phone No. : +82-02-742-3021

References

[1] Minsu Ko, Classification of the symptom severity of mental health records: an approach taking correlations into account using kernel method, 2016 CEGS N-GRID Shared-Tasks and Workshop on Challenges in Natural Language Processing for Clinical Data

[2] Minsu Ko, Sung-Hyon Myaeng, Identifying Disease Definitions with a Correlation Kernel for Symptom Extractions from Text, The 3rd IEEE International Conference on Healthcare Informatics 2014 (ICHI 2014)

[3] H. Jiang, W. Ching, Correlation Kernels for Support Vector Machines Classification with Applications in Cancer Data, Computational and Mathematical Methods in Medicine Volume 2012, Article ID 205025, 7 pages

[4] S. Evert, The Statistics of Word Cooccurrences: Word Pairs and Collocations. Ph.D. thesis. Institut fur maschinelle Sprachverarbeitung, University of Stuttgart, 2005