Constraint Word Clustering Algorithm for Asymmetric Relationship

Rajkumar Jain[1], Narendra S. Chaudhari[2]
[1, 2] Indian Institute of Technology, Indore, 453441, India
[2] Visvesvaraya National Institute of Technology, Nagpur, 440001, India

***Abstract*:** *A new constraint word clustering algorithm is proposed for the given corpus. The proposed method is based on the constraint clustering of words. In this paradigm words are considered similar if they appear in similar contexts and contexts are similar if there word affinity clouds are equivalent. Different types of association between words are identified and based on this association constraints are identified and generated. Proposed constraint algorithm is applicable for words having asymmetric relationship between them; therefore this approach may prove useful as a complement to conventional class-based statistical language modeling techniques.*

Keywords: *Word Clustering, Constraint Clustering, Natural Language Processing, Must-link Constraint.*

1. Introduction

Word Clustering is an important problem in web mining, natural language processing, automatic word classification, word sense, web analytics, computational linguists, and in parsing highly ambiguous syntactic structures [1, 2]. Word clustering [3] is a technique for partitioning sets of words into subsets of similar words. Cluster of words can be identified on the basis of similarity between words or according to the affinities between words. A cluster comprises words that are sufficiently affine with each other. Words in the same cluster are highly affine and words in different cluster are less affine. Word clustering, is a useful approach for improving the performance of sentence retrieval, the more similar the words in each cluster, the better the performance of the retrieval system. Despite the usefulness of word clustering, accurately clustering the words remains a challenging task.

In the context of information retrieval, a new constraint word clustering is projected based on the paradigm of constraints for asymmetric relationship between words. Constraint word clustering approach is appropriate at discovering semantic relationship between words rather than discovering syntactic relationship between words. Affinity [3, 4, 5] describes the quantitative relationship between words. An affinity describes a quantitative relationship between the two words and this in turn helps to identify the clusters of words. A cluster comprises words that are sufficiently affine with each other. A first word is sufficiently affine with a second word if the affinity between the first word and second word satisfies one or more affinity criteria. Present research focuses on the clustering of words based on the finding of semantic relationship between words. Semantic relationships between words are modelled by identifying the constraints. Present research proposes a constraint clustering architecture and algorithm based on the different types of constraint associated between words. Our contribution is summarized as follows: we investigated the constraints based on properties word cloud, we investigated the constraints based on association between words and we presented a constraint word clustering algorithm for asymmetric relationship between words.

2. Related Work

There have been a number of methods proposed in the literature that consider word clustering problem. Words with similar co-occurrence distributions is explored by Brown *et al.*[6], it is based on class-based *n*-gram model in which words are clustered into word classes. Pereira *et al.*[7] present probabilistic membership of words and estimated a soft distributional clustering scheme for certain grammatical co-occurrences. In this strategy the conditional probability of a word is computed by taking advantage of observations of other words that act like this word in this context. A number of variant have been developed on this theme, using

grammatical constraints such as part-of-speech, or morphological units such as lemma, or both [8]. Similarity based model are explored in [9-10] which avoids building clusters. There are algorithms that automatically determine word classes without explicit syntactic or semantic knowledge. In [11] all words are gathered into a single class at the beginning of the procedure, and are successively split to maximize the average mutual information of adjacent classes. In [12], a similar divisive clustering is proposed, based on binomial posteriori distributions on word co-occurrences. Text categorization can be achieved in various ways, in [13] Bag-of-Concepts is used to Improve the Performance of support vector machines. The impact of feature selection on document clustering is discussed in [14]. Hierarchical relationship and associative relationship, is a important in automatically building a thesauri or in finding associative relationship between words. Identification method [15] based on co-occurrence analysis computer the hierarchical relationships between words. Our constraint word clustering method has an advantage over non-constraint clustering algorithm that it extracts background knowledge and guides the algorithm clustering and makes it more suitable for practical use.

3. Affinity Computation and Modelling Based on Co-occurrence

Co-occurrence means coincidence or, frequent occurrence of two terms from a text corpus alongside each other in a certain order. Word co-occurrence in this linguistic sense can be interpreted as an indicator of semantic proximity. The global co-occurrence is an absolute or un-normalized metric. For the purpose of comparing term co-occurrences between different queries and sets of retrieved documents, co-occurrence is normalizing within a practical scale. So, co-occurrence values are normalized in the range of practical scale from 0 to 1.

Definition 1: The affinity [16] between any two words w_a & w_b is defined as the ratio of the number of co-occurrence that include both terms w_a and w_b over the maximum of either the number of co-occurrence contexts that include w_a or the number of co-occurrence contexts that include w_b. The Affinity is given by the following formula:

$$\text{Affinity}(w_a \cap w_b) = \frac{P(w_a \cap w_b)}{\max(P(w_a), P(w_b))} \tag{1}$$

Definition 2: The directional affinity [16] between word w_a & w_b is defined as the conditional probability of observing word w_b, given that word w_a was observed in a co-occurrence context. Directional affinity is used to describe the importance of word of word w_a with respect to word w_b. The directional Affinity (DAffinity) is given by the following formula:

$$\text{DAffinity}(w_a \cap w_b) = \frac{P(w_a \cap w_b)}{P(w_b)} \tag{2}$$

Definition 3: Average directional affinity [16] of a term w_a is the average of the directional affinity of a word with all other words in the co-occurrence contexts. The average directional Affinity (ADAffinity) is given by the following formula:

$$\text{ADAffinity}(w_a) = \frac{\sum_{j=1}^{N} P(w_a \cap w_j)}{N} \tag{3}$$

Definition 4: Differential directional affinity [16] of a term w_a is the difference of directional affinity w_a and average of the directional affinity of word w_a. Differential directional affinity (DiffDaff) is used to normalize the affinity of word with respect to other words.

$$\text{DiffDAff}(w_a) = \text{Affinity}(w_a \cap w_b) - \text{ADAffinity}(w_a) \tag{4}$$

4. Constraint Word Clustering

Definition 5: A word cloud for a word w_a is cloud of words or group of words that are highly with affine with word w_a or all words whose affinity is less than given threshold value δ. A word cloud represents a similarity feature. If clouds of two words are same then they are semantically related. A word cloud for w_a is represented as: $\text{wcloud}(w_a) = \{w_1, w_2, \ldots, w_q\}$, where $\text{DAffinity}(w_a, w_i) \geq \delta$, $1 \leq i \leq q$

4.1 Constraint Modeling

Wagstaff and Cardie [17] introduced constraints in the area of data mining research. Constraints provide guidance about the desired partition and make it possible for clustering algorithms to increase their performance. There are two types of constraints that were termed as must-link constraint and can-not link constraint. In must-link (ML) constraint two instances have to be in the same group, ML(a, b) symbolize instance a and b to have be in the same group. In cannot-link (CL) constraints two instances must not be placed in the same group, CL(a, b) symbolize instance a and b to have be in the different group. Let us consider words w_a and w_b, wcloud(w_a) and wcloud (w_b) are their respective word cloud.

4.1.1 Must Link Constraint

If wcloud(w_a) and wcloud(w_b) are similar then there exist a ML(w_a, w_b) constraint. It is represented in boolean formulation as: $\mathrm{ML}(w_a, w_b) \Rightarrow w_{ak} \wedge w_{bk} = 1$, where, w_{ak} means word w_a belong to k^{th} cluster.

4.1.2 Can-not Link Constraint

If wcloud(w_a) and wcloud (w_b) are not similar then there exist a CL(w_a, w_b) constraint. It can be represented as a boolean formulation as: $\mathrm{CL}(w_a, w_b) \Rightarrow w_{ai} \wedge w_{bi} = 0$ where, w_{ai} means word w_a belong to i^{th} cluster and $i \neq j$.

4.2 Word Clustering Architecture

Constraint word clustering architecture contains two main components: knowledge matrix component and clustering component (See Figure 1). In this architecture, knowledge matrix component facilitates the building of affinity knowledge matrix based on the characteristics of source corpus. By varying the similarity criteria/measure different affinity knowledge matrix can be generated according to the need. Clustering component identify and generate the constraints and produces word cluster. Description of word clustering architecture is as follows:

4.2.1 Indexer

To search large amounts of text quickly it is required to convert the text into a suitable format that allows searching text rapidly. For this purpose a suitable data structure inverted index table is used. Indexer is the component that builds the inverted index table from the source corpus. It eliminates the slow sequential scanning process of the text. Given a corpus $C = \{D_1, D_2, \ldots, D_p\}$ containing p text documents, indexer takes this corpus as an input, identifies the dictionary terms, eliminates the stop words and builds the inverted index table T.

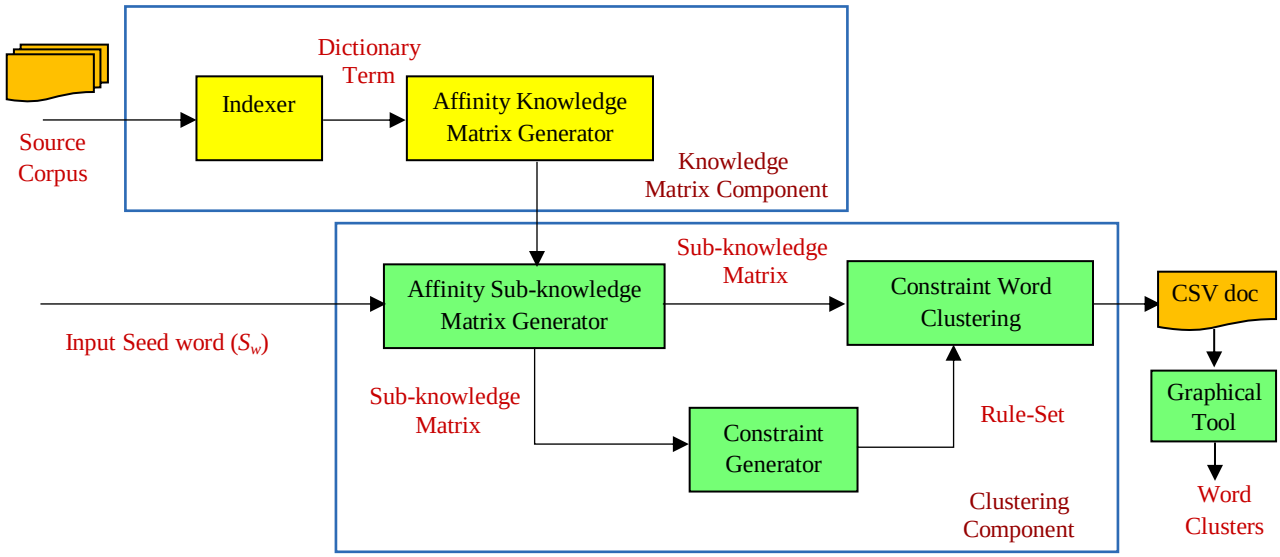

Fig. 1: Architecture of Constraint Word Clustering

4.2.2 Affinity Knowledge Matrix Generator

Affinity knowledge matrix generator builds an affinity knowledge matrix. It is called knowledge matrix because it contains knowledge of whole corpora. It contains information that identifies how two words are closely associated. If inverted index table *T* contains information about *N* words, then corpus matrix generator build a matrix of size *N* * *N*. It can be called as global knowledge affinity matrix corresponding to corpora *C*. It takes inverted index table *T* as the input; find out the affinity between each pair of every term of inverted index table. If size of inverted index table *T* is *N* terms then corpus matrix generator generates an *N* * *N* directional affinity knowledge matrix. Directional Affinity between each pair of term is calculated using (2). Affinity knowledge matrix represents the knowledge of corpora; it represents how the words are inter-related, degree of closeness between words.

4.2.3 Affinity Sub-knowledge Matrix Generator

This component takes input as a seed word and find out the cloud of words which are highly affine with seed word. Let us assume that size of word cloud is *n*. For each word of the cloud there respective word clouds are generated. From this word cloud sub-knowledge matrix is generated of the size *n* * *n*.

4.2.4 Constraint Generator

There are mainly two types of constraints Must Link (ML) constraint and Can Not (CL) constraint. If two objects are associated with ML constraint then they will belong to same group or class. On the other hand if two objects are associated with CL constraint then they will belong to different group or class. Constraint generator performs analysis of sub-knowledge matrix and find out the word cloud which are exactly similar. If the directional average affinities of two clouds are equal then two clouds are having same properties and they have same affinity behavior. If two word clouds are similar then there exist a must link constraint between them. Must link constraint between any two words, enforce that the two words belong to the same cluster. Based on the analysis of knowledge matrix and word cloud following constraints are investigated.

4.2.5 Constraint Word Clustering Algorithm

Constraint word clustering algorithm find words which are highly affine with seed words and cluster them in such a manner that words in the same cluster are highly affine and words in the different cluster are less affine. In the proposed algorithm, existing *k*-means clustering algorithm is modified using concept of constraints. In the proposed constraints word clustering algorithm there are two major modifications, first modification is that words are assigned to the centroid (clusters) according to constraints and affinity values. Second modification is that centroid of cluster is one word it not mean of the cluster like in *k*-means. In the word assignment step of constraint word clustering algorithm, if a word that belongs to RuleML and assigned to cluster C_i, then all others words of RuleML are also moved to cluster C_i. Similarly, if a word that belongs to RuleCL and assigned to cluster C_i, then all others words of RuleCL will not assigned to cluster C_i. Hence constraint clustering word algorithm gives good quality of clusters.

4.3 Investigation and Generation of Constraints and Rulesets

4.3.1 Investigation of Properties and Constraints Generation Based on Word Cloud

Following word cloud properties are investigated:

Property 1. Symmetric Property

if wcloud(w_i) = wcloud(w_j) then wcloud(w_j) = wcloud(w_i) then it gives constraint: ML(w_i,w_j)

Property 2. Transitive Property

if wcloud(w_i) = wcloud(w_j) and wcloud(w_j) = wcloud(w_k) then wcloud(w_i) = wcloud(w_k) then it gives constraint: ML(w_i,w_j, w_k)

Property 3. Implicative Property

if wcloud(w_i) = wcloud(w_j) and if wcloud(w_i) = wcloud(w_k) then wcloud(w_j) = wcloud(w_k) then it gives constraint: ML(w_i,w_j, w_k)

http://dx.doi.org/10.17758/R615506

4.3.2 Constraint Generation Based on Association between Words

Two words are said to be associated if they are having some affinity value between them. Words are associated in either in one direction (forward or backword) or in both direction or not at all. Constratints are investigated on the basis of association between words as follows:

1. *Weak Association*(*One way association*): In weak association either the word w_a is associated with w_b or word w_b is associated with w_a. In weak association induces can not link constraint : CL(w_a, w_b)

2. Strong Association(*Two way association*): In strong association w_a is associated with w_b and word w_b is also associated with w_a. Mathematically, Aff(w_a, w_b) $\neq 0$ and Aff(w_a, w_b) $\neq 0$, it induces no constraint.

3. Zero Association(*no association*): In strong association w_i is not associated with w_j and word w_j is also not associated with w_i. Mathematically, Aff(w_a, w_b) = 0 and Aff(w_a, w_b) = 0. Zero Association induces can not link constraint in forward as well as in backward direction: CL(w_a, w_b) and CL(w_b, w_a)

4.3.3 Rule Set for Generation for ML and CL Constraints

In this Rule set are generated from the ML and CL constraint, rule set guide the constraint word clustering to obtain the desired partition.

4.3.3.1 Rule Set for Generation for ML Constraints

If w_a is the common word between any two ML constraints, then ML constraint can be merged to form a rule of must link constraint called as RuleML.

if ML_1= ML(w_a, w_b) and ML_2= ML(w_c, w_a) then Merge(ML_1, ML_2) => RuleML(w_a, w_b, w_c) = $w_{ai} \vee w_{bi} \vee w_{ci}$ = 1

In general if a RuleML(w_1, w_2,…w_l) contains l words then it is given by the boolean formula:

$$w_{1k} \vee w_{2k} \vee \ldots \vee w_{lk} = 1$$, where w_{lk} means word w_l belong to k^{th} cluster.

4.3.3.2 Rule Set for CL Constraints

If w_a is the common word between any two CL constraints, then CL constraint can be merged to form a rule of can-not link constraint called as RuleCL. if CL_1= CL(w_a, w_b) and CL_2= CL(w_c, w_a) then Merge(CL_1, CL_2) => RuleCL(w_a,w_b,w_c) = $w_{ai} \wedge w_{bj} \wedge w_{ck}$ = 1, where $i \neq j \neq k$. In general if a RuleCL(w_1, w_2,…w_l) contains l words then it is given by the boolean formula: $w_{1k} \wedge w_{2k} \wedge \ldots \wedge w_{lk} = 0$, where w_{lk} means word w_l belong to k^{th} cluster and each word belongs to different cluster.

4.4. Algorithms

Algorithm 1: Affinity Knowledge Matrix

Input: A Corpus C consisting of documents such that $C = \{D_1, D_2, \ldots, D_p\}$, each document D is set of words.
Output: Affinity Knowledge Matrix (AKM) of size $N * N$, where N is the number of words and Inverted Index Table T.

1) Indexing of source corpus C to output inverted index table T, size of table T is N.
2) for i = 1 to N
3) for j = 1 to N
4) $$AKM[i][j] = DAff(w_i \cap w_j) = \frac{P(w_i \cap w_j)}{P(w_j)} \quad (1)$$

Explanation of Algorithm 1: In step 1, source corpus C is indexed and inverted index table T is created. In step 2-4 affinity between each pair of word is calculated and a affinity knowledge matrix is generated.

http://dx.doi.org/10.17758/R615506

Algorithm 2: Word Clustering

Input: Seed word – S_w, Threshold affinity value – δ, number of clusters – k, Affinity Knowledge Matrix- A*KM*, Inverted Index Table – *T*.

Output: k cluster of words

1) if $S_w \in T$ then
2) C_w = WordCloud(AKM, S_w , δ) // C_w is the set of n words such that $C_w = \{w_1, w_2,.,.,w_n\}$
3) for i = 1 to n
4) Wv_i = WordCloud(AKM, w_i)
5) for i = 1 to n
6) for j = 1 to sizeof(Wv_i)
7) if $w_{ij} \notin C_w$ then delete w_i from Wv_i. (w_{ij} stands for j^{th} word of i^{th} wordcloud of Wv_i)
8) ASKM = AffintySubKnowledgeMatrix(Wv_1, Wv_2, . , . , Wv_n)
9) Analyze ASKM, Let MLC = {ml_1, ml_2, ., ., ., ml_q} are q ML constraints are investigated (refer)
10) $\forall\, i, l, m,$ if $(w_i \in ml_l)$ and $(w_i \in ml_m)$ then (where $1 \le i \le n, 1 \le l, m \le q$)
11) $rulelist = merge(ml_l, ml_m)$
12) Assign k words as centroids c = {c_1, c_2, ., ., ., c_k} of k clusters C= {C_1, C_2, ., ., ., C_k}
13) for i = 1 to k
14) $C_i = \{w_j : \max\left(\text{aff}\left((w_j, c_i)\right)\right) \forall\, j, 1 \le j \le n\}$ // Assignment step
15) if $\left(w_j \in mlrulellist\right)$ then
16) $C_i = \{w_r : (w_r \in mlrulelist)\ \forall\ w_r\}$ // Assignment on the basis of constraints.
17) do the following until old centroids and new centroids are same
18) for i = 1 to k
19) η =sizeof(C_i)
20) for j = 1 to η
21) $\hat{W}v_j$ = WordCloud(SKM, $\hat{w}_j$) where, $(\hat{w}_j \in C_i)$
22) for l = 1 to sizeof($\hat{W}v_j$)
23) if $(w_{il} \notin C_w)$ then delete w_i from $\hat{W}v_j$. (w_{il} stands for l^{th} term of i^{th} word cloud of $\hat{W}v_j$)
24) CKM = ClusterSubKnowledgeMatrix($\hat{W}_{v1}$, $\hat{W}_{v2}$, . , . , $W_{v\eta}$)
25) for j = 1 to η
26) $AA(w_j) = \sum_{m=1}^{\eta} CKM\,[j][m]/\eta$
27) $MeanCluster(C_i) = \forall w_j : (w_j \in C_i) \sum_{j=1}^{\eta} AA(w_j)/\eta$

Explanation of Algorithm 2: In step 1 it is checked whether the input seed word belong to corpus or not. In step 2, word cloud is generated corresponding to seed word S_w. C_w contains all words which are affine with S_w and whose affinity value is less than δ. In step 3-4, word clouds are generated for all n belonging to C_w.. In step 5-7, spurious words are deleted from each word cloud, spurious words are that word which does not belong to C_w or which are not affined with S_w. This step normalizes the size of each word cloud to n. In step 8, Sub knowledge matrix is generated from n word cloud of size $n * n$. In step 9, Sub knowledge matrix is analyzed and ML constraints are generated. In step 10-11, if a word belongs to more than one ML constraints then all ML constraints are merged to form a rulelist. If ML constraints are not mutually related then, they ML constraints will belong to different rulelists. In step 12, randomly k words are assigned as centroids of k-clusters. In step 13-14 words are assigned to their respective clusters based on their maximum affinity with the centroids. In step 15-16, if a word w_r is an element of to a *mlrulelist* and belongs to a cluster C_i then all other words of same rule list are assigned to cluster C_i and their status is updated to assigned so assigned words are not checked for comparison and assignment. Step 18-28 is executed until there is no change in the centroids of two consecutive iterations. In step 18-24, for each word of each cluster, word cloud is generated and after removal of spurious words Cluster-knowledge matrix is generated. In step 25-26 average affinity of each word in cluster is

http://dx.doi.org/10.17758/R615506

calculated. In step 27, new mean of each cluster is calculated. In step 28, word w_j which is closest to the mean of the cluster C_j is assigned as the new centroid cluster C_j.

5. Concluding Remarks

We proposed a method to identify and generate constraints between words that identify semantic similarity measure between words and word clouds. We investigated different types of association between words and identified constraints based on the investigated association between words. Moreover, a constraint based word clustering algorithm is proposed. In the proposed approach, words clouds are compared rather than words, which extract semantic meaning of words in the respective group of words or in the respective affinity sub-knowledge matrix for the generation of constraints. Constraints provide guidance about the desired partition and make it possible for clustering algorithms to increase the accuracy of clusters generated. Proposed approach is applicable for symmetric as well as asymmetric relationship between words. Thus, constraint word clustering algorithm is useful extension to conventional word clustering algorithm.

6. References

[1] Dagan, L. Lee, F. C. N. Pereira, "Similarity Based Model Of Word Co-Occurrence Probabilities", *Machine Learning - Special issue on natural language learning archive*, Kluwer Academic Publishers Hingham, MA, USA, Vol. 34, 1999, pp. 43 – 69.

[2] D. Bollegala, Y. Matsuo, M. Ishizuka, "A Web Search Engine-Based Approach to Measure Semantic Similarity between Words", *IEEE Transactions On Knowledge And Data Engineering*, Vol. 23(7), 2011, pp. 977-990. http://dx.doi.org/10.1109/TKDE.2010.172

[3] H. Li, N. Abe, "Word Clustering and Disambiguation Based on Co-occurrence Data", *Journal of Natural Language Engineering*, Vol 8(1),2002, pp. 25 – 42 http://dx.doi.org/10.1017/S1351324902002838

[4] Information Retrieval, C. J. Van Rijsbergen, Butterworth-Heinemann; 2nd edition (March 1979)

[5] Y. Karov, S. Edelman. "Learning similarity-based word sense disambiguation from sparse data", *In Proceedings of the Fourth Workshop on Very Large Corpora*, 1996, pp 1-18.

[6] P. F. Brown, V. J. DellaPietra, P. V. deSouza, J. C. Lai, R. L. Mercer, "Class-based *n*-gram models of natural language". *Computational Linguistics*, Vol.*18*(4), 1992, pp.467-479.

[7] F. C. N. Pereira, N. Tishby, L. Lee, "Distributional clustering of English words", *In 31st Annual Meeting of the Association for Computational Linguistics*, Somerset, New Jersey, 1993, pp. 183-190. http://dx.doi.org/10.3115/981574.981598

[8] G. Maltese , F. Mancini, "An Automatic Technique to Include Grammatical and Morphological Information in a Trigram- Based Statistical Language Model," *in Proc. ICASSP*, San Francisco, CA, 1992, pp. 157-160, 1992 http://dx.doi.org/10.1109/icassp.1992.225948

[9] I. Dagan, S. Marcus, S. Markovitch, "Contextual word similarity and estimation from sparse data", *In 31st Annual Meeting of the ACL* ,Somerset, New Jersey, 1993, pp. 164–171. http://dx.doi.org/10.3115/981574.981596

[10] Dagan, I., Marcus, S., & Markovitch, S. "Contextual word similarity and estimation from sparse data".*Computer Speech and Language*,1995, Vol. 9, 123–152. http://dx.doi.org/10.1006/csla.1995.0008

[11] M. Jardino and G. Adda, "Automatic Word Classification, Using Simulated Annealing", *in Proc.1993 ICASSP*, Minneapolis, MN,1993, pp. 41-44.

[12] M. Tamoto and T. Kawabata, "Clustering Word Category Based on Binomial Posteriori Co-Occurrence Distribution", *in Proc. 1995 ICASSP*, Detroit, MI, 1995, pp. 165-168.

[13] Sahlgren, Magnus, and Rickard Cöster. "Using bag-of-concepts to improve the performance of support vector machines in text categorization." *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics, 2004. http://dx.doi.org/10.3115/1220355.1220425

[14] Y. Liu, X. Wang, B. Liu, "A Feature Selection Algorithm for Document Clustering based On Word Co-Occurrence Frequency", *in Proceedings of the Third International Conference on Machine Learning and Cybernetics*, Shanghai, 2004, pp 2963-2968

[15] W. Zhou, Y. Du, H. Wang, X Lv, "Automatic identification of hierarchical relationship between words based on clustering", *In International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), 2011*, pp. 1585-1588. http://dx.doi.org/10.1109/TMEE.2011.6199512

[16] D.L. Marvit, J. Jain, S. Stergiou, A. Gilman, B.T. Adler, J.J. Sidorowich, "Identifying Clusters of Words According To Word Affinities", *Google Patents*, 2009,12/242,957.
[17] K. Wagsta, C. Cardie , "Clustering with Instance-level Constraints", *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 1103-1110.