A Fast Approximate AIB Algorithm for Distributional Word Clustering

Lei Wang, Jianjia Zhang†, Luping Zhou, Wanqing Li
School of Computer Science and Software Engineering
University of Wollongong, Wollongong, NSW, Australia, 2522
leiw,lupingz,wanqing@uow.edu.au, jz163@uowmail.edu.au†

Abstract

Distributional word clustering merges the words having similar probability distributions to attain reliable parameter estimation, compact classification models and even better classification performance. Agglomerative Information Bottleneck (AIB) is one of the typical word clustering algorithms and has been applied to both traditional text classification and recent image recognition. Although enjoying theoretical elegance, AIB has one main issue on its computational efficiency, especially when clustering a large number of words. Different from existing solutions to this issue, we analyze the characteristics of its objective function — the loss of mutual information, and show that by merely using the ratio of word-class joint probabilities of each word, good candidate word pairs for merging can be easily identified. Based on this finding, we propose a fast approximate AIB algorithm and show that it can significantly improve the computational efficiency of AIB while well maintaining or even slightly increasing its classification performance. Experimental study on both text and image classification benchmark data sets shows that our algorithm can achieve more than 100 times speedup on large real data sets over the state-of-the-art method.

1. Introduction

Distributional word clustering originates from text classification. It was first proposed in [3, 16] for word classification and then developed by [1] for document classification. The work in [1] summarizes the three key benefits of distributional word clustering as discovering semantic word groups, producing compact classification models and improving classification performance. Other work on studying distributional word clustering for text classification can be found in [2, 4]. Most, if not all, of the existing distributional word clustering algorithms use an objective function that can be explicitly or implicitly linked to the criterion of mutual information. The Agglomerative Information Bottleneck (AIB) [17] is a typical one of them and has been widely applied. Rooted in the Information Bottleneck (IB) framework, AIB conducts word clustering by maximally preserving the mutual information between words and class labels. It works in a bottom-up hard hierarchical clustering way. Starting with each word as an individual cluster, AIB merges two words leading to the minimum loss of mutual information at each level. As argued in [17], compared to the top-down methods, such as that in [16], AIB generates fully deterministic clustering result for any desired number of clusters in one shot. Meanwhile, it is pointed out in [17] that the main disadvantage of AIB is the high computational load. Its time complexity is $\mathcal{O}(n^3)$ and space complexity is $\mathcal{O}(n^2)$, which prevent it from scaling well for large sets of words.

In the past several years, the bag-of-words model has been introduced from text analysis to generic image recognition and achieved high success. Accordingly, efficiently merging a large set of visual words to generate compact visual codebooks has attracted attention. A number of methods have recently been developed [20, 11, 7, 19, 12, 9, 14], among which AIB has been successfully used in [11, 12, 7]. In particular, to make AIB scalable to a large number of words, say, 10000, the work in [7] proposes a fast implementation of AIB (called Fast-AIB in this paper). It reduces the time and space complexity of AIB by one order to $\mathcal{O}(n^2)$ and $\mathcal{O}(n)$, respectively.

In this paper, we further improve the computational efficiency of AIB by developing a fast *approximate* algorithm. By analyzing the function of mutual information loss, we find that two words having more similar ratios of word-class joint probabilities usually incur smaller loss, and prove that the best pair of words selected in this way is indeed optimal when only the ratio information of each word is available. Inspired by this finding, we propose an approach different from AIB and Fast-AIB, which evaluate the goodness of every pair of words by explicitly computing the mutual information loss. Instead, our algorithm evaluates the goodness of a pair of words by merely matching their ratios of word-class joint probabilities. And an explicit evaluation of mutual information loss is only used to select the

1063-6919/13 $26.00 © 2013 IEEE
DOI 10.1109/CVPR.2013.78

best pair from a small set of good candidate word pairs. Also, once two words are merged, our algorithm can trivially obtain the good candidate words for the next level. By doing so, we can further reduce the time complexity of AIB to $\mathcal{O}(n \log n)$ while keeping the space complexity at $\mathcal{O}(n)$, making it more efficient in handling a large number of words such as tens of thousands. Moreover, our algorithm can well maintain the classification performance of AIB (and Fast-AIB) and even slightly improve it when dealing with a very large set of words. By looking into this result, we think that this could be because the approximation in our algorithm, which only requires the information loss to be small rather than rigidly minimum as in AIB, produces a mild "regularization" effect. This seems to help enhance the robustness of AIB with respect to less reliable probability estimates, when the number of words is much larger than that of training samples. To verify the advantages of our algorithm, experimental study is conducted on three benchmark data sets on image recognition and text classification, with different number of words involved. As demonstrated, compared with Fast-AIB, our algorithm can further improve its speed by more than 100 times on large real data sets, while achieving comparable or slightly higher classification performance.

2. Background and related work

2.1. Agglomerative Information Bottleneck [17]

Let $\mathcal{W} = \{w_1, w_2, \cdots, w_n\}$ denote a set of n words and let c be class labels, where $c = 1, \cdots, C$. The mutual information between $\mathcal{W}$ and c is denoted by $I(\mathcal{W}, c)$. Assuming that two words w_p and w_q are merged into a new word w_{pq}, the word set becomes $\mathcal{W}_{pq} = \{\mathcal{W} \setminus \{w_p, w_q\}\} \bigcup \{w_{pq}\}$ and the mutual information turns to be $I(\mathcal{W}_{pq}, c)$ accordingly. This incurs the loss of mutual information as

$$\Delta I(w_p, w_q) \quad = \quad I(\mathcal{W}, c) - I(\mathcal{W}_{pq}, c). \tag{1}$$

By merging, the prior probability and the word-class joint probability of the new word w_{pq} are defined as

$$\begin{aligned} P(w_{pq}) &= P(w_p) + P(w_q) \\ P(w_{pq}, c) &= P(w_p, c) + P(w_q, c). \end{aligned} \tag{2}$$

A basic implementation of AIB in [17] pre-computes all possible $\Delta I(w_p, w_q)$ for $1 \le p < q \le n$ and saves it as a matrix $\mathbf{D}$. At each level l, the minimum ΔI is sought to find the optimal word pair (w_p, w_q) to generate a new word w_{pq}. After that, the clustering proceeds to the next level $(l-1)$, where $\Delta I(w_{pq}, w_i)$ is updated for all the remaining words w_i (corresponding to updating one row and one column of $\mathbf{D}$), and ΔI for other pairs of words is left unaffected. Searching for the minimum ΔI and a new optimal pair of words is then repeated. With this implementation, completely hierarchically merging n words incurs time complexity $\mathcal{O}(n(n^2+nC))$ and space complexity of $\mathcal{O}(n^2)$. This causes a lengthy merging process and significant memory cost when n is large.

2.2. Fast-AIB [7]

The work in [7] proposes a smart modification of the above basic implementation and significantly improves the computational efficiency. The Fast-AIB method still precomputes $\Delta I(w_p, w_q)$ for all possible pairs of words. However, instead of saving the whole matrix $\mathbf{D}$, it finds the minimum ΔI along the ith row of $\mathbf{D}$ and records the index of the corresponding word and this minimum value as $(k_i, \Delta I(w_i, w_{k_i}))$. This not only reduces the time complexity of searching for the minimum ΔI over $\mathbf{D}$ to $\mathcal{O}(n)$ at each level, but also decreases the space complexity to $\mathcal{O}(n)$. When two words w_p and w_q are merged, Fast-AIB will firstly update the entries $(k_i, \Delta I(w_i, w_{k_i}))$ for which $k_i = p$ or q. After that, the remaining entries will be checked whether they need to be updated due to the presence of the new word w_{pq}. This can be quicky implemented by checking whether $\Delta I(w_i, w_{k_i})$ is larger than $\Delta I(w_i, w_{pq})$. According to [7], the total time complexity of Fast-AIB reduces to $\mathcal{O}(n^2)$ and the space complexity becomes $\mathcal{O}(n)$ only.

3. The proposed method

3.1. Analysis of information loss function ΔI

Given a set of training samples, the mutual information between $\mathcal{W}$ and c can be computed by

$$I(\mathcal{W}, c) = \sum_c \sum_{i=1}^{n} P(w_i, c) \log \frac{P(w_i, c)}{P(w_i)P(c)}, \tag{3}$$

where $P(w_i, c)$ is the word-class joint probability of w_i and c, and $P(w_i)$ and $P(c)$ are the prior probabilities. The loss of mutual information can be calculated as

$$\begin{aligned} \Delta I(w_p, w_q) &= I(\mathcal{W}, c) - I(\mathcal{W}_{pq}, c) \\ &= \sum_c \left(\sum_{i=p,q} P(w_i, c) \log \frac{P(w_i, c)}{P(w_i)P(c)} \right. \\ &\qquad \left. - P(w_{pq}, c) \log \frac{P(w_{pq}, c)}{P(w_{pq})P(c)} \right), \end{aligned} \tag{4}$$

where w_{pq} is the new word generated by merging words p and q. In this paper, we focus on binary classification and define $c = \pm 1$. We can fully represent $\Delta I(w_p, w_q)$ with four terms $P(w_p, +1)$, $P(w_p, -1)$, $P(w_q, +1)$ and $P(w_q, -1)$. Now we give **important definitions** that will be used throughout the following parts,

$$a_i \triangleq P(w_i, +1); \quad b_i \triangleq P(w_i, -1); \quad r_i = b_i / a_i; \tag{5}$$

where $i = 1, \cdots, n$ and $a_i, b_i > 0$ is assumed [1]. When necessary, we will write word w_i in full as $w_i(a_i, b_i)$. Note that r_i denotes the "class ratio" for each word, and it will play a fundamental role in our algorithm.

With the above definitions, $\Delta I(w_p, w_q)$ is expressed as

$$\begin{aligned}\Delta I(w_p, w_q) &= \Delta I(a_p, b_p, a_q, b_q) \qquad (6)\\ &= a_p \log\left[\frac{a_p}{a_p + a_q}\left(1 + \frac{a_q + b_q}{a_p + b_p}\right)\right]\\ &+ b_p \log\left[\frac{b_p}{b_p + b_q}\left(1 + \frac{a_q + b_q}{a_p + b_p}\right)\right]\\ &+ a_q \log\left[\frac{a_q}{a_p + a_q}\left(1 + \frac{a_p + b_p}{a_q + b_q}\right)\right]\\ &+ b_q \log\left[\frac{b_q}{b_p + b_q}\left(1 + \frac{a_p + b_p}{a_q + b_q}\right)\right].\end{aligned}$$

As seen, $\Delta I(w_p, w_q)$ is a symmetrical function with respect to (a_p, b_p) and (a_q, b_q). Swapping them does not change this function. This can be intuitively understood because merging operation is symmetrical with respect to two words. We can therefore fix (a_p, b_p) and scrutinize the characteristics of this function with respect to (a_q, b_q).

Now we show the connection between the minimization of ΔI and the relationship of (a_p, b_p) to (a_q, b_q).

Theorem 1. *For given a_p, b_p, the function of mutual information loss in Eq. (6) reduces to $\Delta I(a_q, b_q)$. It can be proved that*

$$\begin{cases} \frac{\partial \Delta I(a_q, b_q)}{\partial a_q} = 0 \\ \frac{\partial \Delta I(a_q, b_q)}{\partial b_q} = 0 \end{cases} \iff \frac{b_q}{a_q} = \frac{b_p}{a_p} \qquad (7)$$

Proof: From Eq. (6), it can be obtained that

$$\frac{\partial \Delta I(a_q, b_q)}{\partial a_q} = \log \frac{a_q(a_p + b_p + a_q + b_q)}{(a_p + a_q)(a_q + b_q)}. \qquad (8)$$

It is not difficult to verify that

$$\frac{a_q(a_p + b_p + a_q + b_q)}{(a_p + a_q)(a_q + b_q)} = 1 \iff \frac{b_q}{a_q} = \frac{b_p}{a_p}. \qquad (9)$$

This proves

$$\frac{\partial \Delta I(a_q, b_q)}{\partial a_q} = 0 \iff \frac{b_q}{a_q} = \frac{b_p}{a_p}. \qquad (10)$$

Similarly, it can be obtained that

$$\frac{\partial \Delta I(a_q, b_q)}{\partial b_q} = 0 \iff \frac{b_q}{a_q} = \frac{b_p}{a_p}. \qquad (11)$$

[1] This can easily be satisfied because in practice a small positive number is usually appended to each data entry in order to obtain reliable estimates of the probabilities.

Combing Eq. (10) and Eq. (11) completes the proof. ■

This result indicates that for a given (a_p, b_p), any (a_q, b_q) satisfying $b_q/a_q = b_p/a_p$ must be a stationary point, at which $\Delta I(a_q, b_q)$ takes an extreme value. And for any stationary point of $\Delta I(a_q, b_q)$, it must satisfy $b_q/a_q = b_p/a_p$. Moreover, it is easy to see that

$$\Delta I(a_q, b_q)\big|_{\frac{b_q}{a_q} = \frac{b_p}{a_p}} = 0. \qquad (12)$$

This indicates that at all stationary points, the function ΔI indeed achieves its minimum value of zero, that is, no loss of mutual information. Geometrically, all stationary points (a_q, b_q) reside on a straight line $b_q - r_p a_q = 0$, where $r_p = b_p/a_p$ is the class ratio of word w_p. Figure 1 illustrates this case. Therefore, the pair of words that leads to *exact zero* loss can be easily identified by simply comparing the class ratios of the two words, that is, whether $r_p = r_q$ or not. However, this result is not as helpful as its first glance, because each word usually does not have an identical class ratio in practice. So, the goodness of a word pair may have to be evaluated by explicitly computing the incurred information loss ΔI, as done in AIB and Fast-AIB.

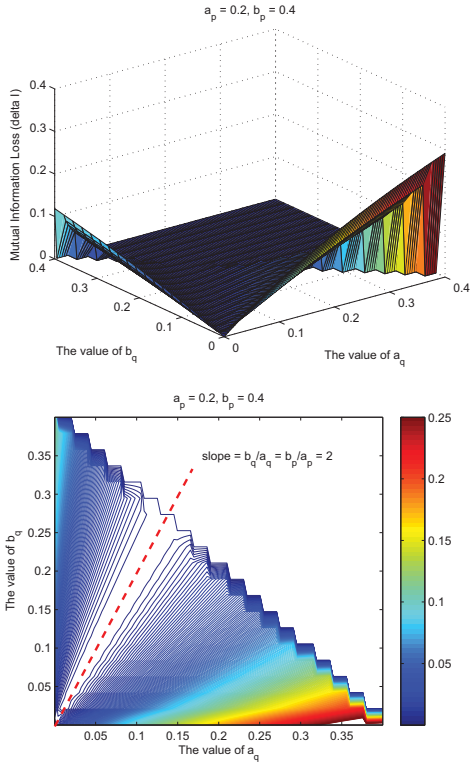

Figure 1. The graph of $\Delta I(a_q, b_q)$ for a given (a_p, b_p). The top is side view and the bottom is top view. The dashed red line, $b_q - r_p a_q = 0$, indicates the points where the minimum value of zero is achieved.

Nevertheless, the above result motivates us to consider the following question: *in addition to identifying the word*

pair incurring exact zero information loss, whether a simple comparison of class ratios can be used to suggest good candidate word pairs to merge? If yes, this would bring significant computational advantage over the existing methods, where the incurred loss has to be explicitly evaluated for every possible pair of words to identify the optimal pair. To answer this question, we examine the relationship between $|r_i - r_j|$ and $\Delta I(w_i, w_j)$. In specific, will a smaller $|r_i - r_j|$ lead to less $\Delta I(w_i, w_j)$? In the following, we use Theorem 2 and Corollary 1 to show: *when only using the class ratio information of each word*, for any given word w_i, merging it with the word w_j having the smallest $|r_i - r_j|$ is indeed optimal. The optimality is in the sense of minimizing the "maximum" information loss over all possible word $w_j(a_j, b_j)$ for which $b_j/a_j = r_j$.

Theorem 2. Let w_i be a given word with class ratio r_i. w_j is another word with class ratio r_j, where $r_i \leq r_j \leq r_0$ and r_0 is a constant. No other information on the two words is accessible except r_i and r_j. Let $\Delta I_{max}(w_i, w_j)$ be the maximum information loss caused by merging w_i to w_j,

$$\Delta I_{max}(w_i, w_j) \triangleq \max_{w_j \in \mathcal{S}} \Delta I(w_i, w_j), \tag{13}$$

where the set $\mathcal{S} = \{w_j(a_j, b_j) | r_i \leq r_j \leq r_0;\ a_j + b_j \leq 1\}$. Then $\Delta I_{max}(w_i, w_j)$ is achieved only when $r_j = r_0$. In addition, this conclusion is also true when $r_i \geq r_j \geq r_0$.

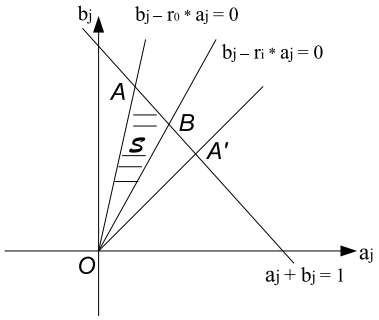

Figure 2. Illustration of the set $\mathcal{S}$ in Theorem 2.

Proof: Let us consider $r_i \leq r_j \leq r_0$ first. This problem is illustrated in a 2D coordinate system $a_j O b_j$ in Figure 2. The set $\mathcal{S}$ corresponds to the triangle ΔAOB, which is the intersection of three half-spaces in the first quadrant:

$$b_j - r_i a_j \geq 0; \quad b_j - r_0 a_j \leq 0; \quad a_j + b_j \leq 1. \tag{14}$$

Note that no tighter upper bound can be set for the last inequality because only the class ratio of each word is accessible. To prove this theorem is equivalent to proving that $\Delta I_{max}(w_i, w_j)$ is achieved only on the line segment OA. Let $\mathbf{g} = \left(\frac{\partial \Delta I}{\partial a_j}, \frac{\partial \Delta I}{\partial b_j}\right)^\top$ be the gradient of $\Delta I(w_i, w_j)$ with respect to a_j and b_j. Recalling the proof of Theorem 1, it is known that for any stationary point of $\Delta I(w_i, w_j)$, there is

$$\mathbf{g} = \mathbf{0} \Longleftrightarrow \frac{b_j}{a_j} = \frac{b_i}{a_i} \Longleftrightarrow r_j = r_i \tag{15}$$

and at this time $\Delta I(w_i, w_j)$ achieves its minimum value of zero. This indicates that except on OB, there is no stationary point in the region $\mathcal{S}$. Hence, $\Delta I_{max}(w_i, w_j)$ can only be achieved on the boundary, that is OA or AB.

Now we rule out the possibility of AB by showing that $\Delta I(w_i, w_j)$ monotonically increases along the line segment $\overrightarrow{BA}$. Noting that the direction of $\overrightarrow{BA}$ is $\mathbf{v} = (-1, 1)^\top$, the directional derivative of $\Delta I(w_i, w_j)$ along $\overrightarrow{BA}$ is

$$\mathbf{g}^\top \mathbf{v} = \log \frac{a_j(b_i + b_j)}{b_j(a_i + a_j)} \geq 0, \tag{16}$$

where "$\geq$" is achieved because $r_j \geq r_i$ is given. Hence, the maximum value of $\Delta I(w_i, w_j)$ on AB must be achieved at point A. This concludes that $\Delta I_{max}(w_i, w_j)$ can only be achieved on OA, which corresponds to w_j with $r_j = r_0$.

When the condition becomes $r_i \geq r_j \geq r_0$, the above proof can still be applied except that in this case

$$\mathbf{g}^\top \mathbf{v} = \log \frac{a_j(b_i + b_j)}{b_j(a_i + a_j)} \leq 0, \tag{17}$$

where "$\leq$" is achieved because $r_j \leq r_i$ is given. Hence, the maximum value of $\Delta I(w_i, w_j)$ on BA' must be achieved at point A'. This concludes that $\Delta I_{max}(w_i, w_j)$ can only be achieved on OA', which corresponds to w_j with $r_j = r_0$. ■

Corollary 1. Let $\mathcal{L}_1 = \{w_1, w_2, \cdots, w_n\}$ be a list of n words, which have been sorted such that their class ratios satisfy $r_1 \leq r_2 \leq \cdots \leq r_n$. For any given word w_i, in order to minimize $\Delta I_{max}(w_i, w_j)$ the optimal word w_j must be either w_{i+1} or w_{i-1}, one of the two neighbors of w_i in $\mathcal{L}_1$.

Proof: It is proved by contradiction. Assuming that the optimal word is w_{i+k} with $k > 1$, it can be known from Theorem 2 that

$$\begin{aligned} \max_{w_{i+k} \in \mathcal{S}_1} \Delta I(w_i, w_{i+k}) &= \max_{w_j \in \mathcal{S}_2} \Delta I(w_i, w_j) \\ &\geq \max_{w_{i+1} \in \mathcal{S}_3} \Delta I(w_i, w_{i+1}), \end{aligned} \tag{18}$$

where $\mathcal{S}_1 = \{w(a, b) | r = r_{i+k}, a + b \leq 1\}$ contains all words having the class ratio of r_{i+k}; $\mathcal{S}_2 = \{w(a, b) | r_i \leq r \leq r_{i+k}, a + b \leq 1\}$ contains all words having the class ratio between r_i and r_{i+k}; and $\mathcal{S}_3 = \{w(a, b) | r = r_{i+1}, a + b \leq 1\}$ contains all words having the class ratio of r_{i+1}. Note that the "$\geq$" is obtained because $\mathcal{S}_3 \subset \mathcal{S}_2$. This result means that

$$\Delta I_{max}(w_i, w_{i+k}) \geq \Delta I_{max}(w_i, w_{i+1}). \tag{19}$$

This contradicts with the assumption at the beginning of proof and therefore w_{i+1} shall be a better choice. In a similar way, it can be proved that w_{i-1} shall be a better choice than w_{i-k} when $k > 1$. ■

Through the above analysis, it can be seen that for any given word w_i, we can find an "optimal" word w_j to merge by simply comparing the class ratios of the n words. At the same time, note that the "optimal" is in the sense of minimizing $\Delta I_{max}(w_i, w_j)$. It measures the worst case (i.e., the maximum information loss over all possible w_j with class ratio equal to r_j) rather than the loss actually incurred by merging the two words. This is the price that we pay for only using the class ratios to gain computational advantage. In addition, it becomes difficult to determine in further which one of w_{i-1} and w_{i+1} in Corollary 1 really minimizes $\Delta I_{max}(w_i, w_j)$ by merely based on the class ratio information. In this work we do not pursue this further because it is not in the spirit of our motivation. Instead, after w_{i-1} and w_{i+1} are identified, we will simply compare the loss actually incurred by them and pick the smaller one.

Now we show the last result needed for developing our fast approximate algorithm. It indicates that once two words are merged, the good candidate word pairs for the next level can be trivially obtained.

Theorem 3. Recall that $\mathcal{L}_1 = \{w_1, w_2, \cdots, w_n\}$ is a sorted list satisfying $r_1 \leq r_2 \leq \cdots \leq r_n$. Merging $w_i(a_i, b_i)$ and $w_{i+1}(a_{i+1}, b_{i+1})$ produces a new word w_i'. This operation does not alter the order of words in the list $\mathcal{L}_1$.

Proof: It is known from Eq. (2) that the new word is $w_i'(a_i + a_{i+1}, b_i + b_{i+1})$. It suffices to show that for w_i', its class ratio r_i' satisfies $r_i \leq r_i' \leq r_{i+1}$. This is indeed true because it is trivial to verify that $\forall a_i, b_i, a_{i+1}, b_{i+1} > 0$,

$$\frac{a_i}{b_i} \leq r_i' = \frac{a_i + a_{i+1}}{b_i + b_{i+1}} \leq \frac{a_{i+1}}{b_{i+1}}. \tag{20}$$

This completes the proof. ■

The above result shows that we only need to sort the n words once at the beginning, and the two neighbors of any newly generated word can be trivially identified.

3.2. The proposed algorithm FA-AIB

Now our algorithm, called FA-AIB, is listed as follows. Recall that $\Delta I(w_i, w_j)$ denotes the loss of mutual information actually incurred by merging words w_i and w_j.

1. Given a set of n words $\{w_i(a_i, b_i)\}_{i=1}^n$, sort them based on their class ratios r_i to obtain the ordered list $\mathcal{L}_1 = \{w_1, w_2, \cdots, w_n\}$. According to Corollary 1, $(n-1)$ good candidate word pairs are identified, which are (w_i, w_{i+1}) with $i = 1, 2, \cdots, (n-1)$.

2. Evaluate the value of ΔI for each of the $(n-1)$ pairs, respectively. Sort the $(n-1)$ values of ΔI to obtain a list $\mathcal{L}_2$.

3. Identify the optimal pair (w_i^*, w_{i+1}^*) that corresponds to the minimum ΔI value in $\mathcal{L}_2$. Merge w_{i+1} into w_i and remove w_{i+1} from $\mathcal{L}_1$. Now the length of $\mathcal{L}_1$ decreases from n to $n-1$.

Table 1. Complexity comparison to relevant algorithms

Algorithm	Time	Space
AIB	$\mathcal{O}(n(n^2+2n))$	$\mathcal{O}(n^2)$
F-AIB	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
FA-AIB (proposed)	$\mathcal{O}(n \log n)$	$\mathcal{O}(n)$
FA-AIB-s (simplified)	$\mathcal{O}(n \log n)$	$\mathcal{O}(n)$

4. **Decrease** n **by setting** $n := n - 1$.

5. The new word is $w_i(a_i + a_{i+1}, b_i + b_{i+1})$. As shown by Theorem 3, the order of words in $\mathcal{L}_1$ does not change, and therefore the two neighbors of the new word w_i can be trivially identified. For the convenience of presentation, all the words in $\mathcal{L}_1$ are re-numbered as $w_1, w_2, \cdots, w_n$ by following their order in $\mathcal{L}_1$.

6. Evaluate ΔI for merging the new word with its two neighbors in the list $\mathcal{L}_1$, respectively. Insert the two ΔI values into $\mathcal{L}_2$. When the current number of words, n, is larger than a predefined threshold n_0, go to step 3 and terminate otherwise.

The following analyzes the computational complexity of the proposed algorithm with respect to each step.

1. It sorts n words, which can be done at $\mathcal{O}(n \log n)$ in the average case by quick sort.

2. It evaluates ΔI by $(n-1)$ times, which can be done at $\mathcal{O}(n)$. Sorting the $(n-1)$ values incurs $\mathcal{O}(n \log n)$.

(3,5). The operation can be done at $\mathcal{O}(1)$ in both steps.

6. Evaluating two ΔI values can be done at $\mathcal{O}(1)$. Inserting them into $\mathcal{L}_2$ is at $\mathcal{O}(\log n)$ by binary search.

Considering the loop of steps 3, 4, 5 and 6, the total time complexity of the proposed algorithm will be at the order of $\mathcal{O}(n \log n)$. Also, since this algorithm only needs to store $\{(a_i, b_i)\}_{i=1}^n$ and maintains two lists $\mathcal{L}_1$ and $\mathcal{L}_2$, its space complexity is at $\mathcal{O}(n)$. The comparison to the existing algorithms AIB and Fast-AIB is listed in Table 1.

Discussion. To gain computational advantage, our algorithm identifies good candidate word pairs based on the worst-case loss instead of the loss actually incurred. However, it is observed that this does not necessarily hurt classification performance. Surprisingly, sometimes it could even slightly improve the performance, especially when a large number of words are to be merged. Our explanation is that the approximation in our algorithm may implicitly introduce a mild "regularization" effect, which only requires the loss to be small rather than strictly minimum as in AIB and Fast-AIB. Specifically, when the number of words is large, the occurrence of each word in a set of training samples will become sparse. This could lead to two issues:

1. The joint probabilities estimated via the frequency counts become less reliable. This will in turn affect the reliability of evaluating ΔI on these probabilities;
2. In the case of small sample size, minimizing ΔI based on the limited training data to determine the optimal word pair is prone to overfitting the samples, and the generated new words may not generalize well on test data. These could bring adverse effect to classification.

3.3. FA-AIB-s: A simplified variant of FA-AIB

In addition, we try a simplified variant of FA-AIB and call it FA-AIB-s. Our motivation for FA-AIB-s lies in that *we want to test how far a simple comparison of the class ratios can go in appropriately clustering words*. Instead of selecting the optimal pair (w_i, w_{i+1}) as the one having the minimum ΔI, FA-AIB-s simply selects the optimal pair as the one having the minimum $\Delta r = |r_i - r_{i+1}|$. Evidently, FA-AIB-s uses a much coarser approximation to strictly minimizing ΔI, because as seen from Theorem 1, such a selection only ensures the mergence to be "optimal" in very specific cases. For FA-AIB-s, the order of its computational complexity is the same as FA-AIB. However, its speed in practice will be faster due to completely avoiding calculating ΔI in the whole course. Also, we are interested in checking if FA-AIB-s can still maintain good classification performance and when it will fail.

4. Experimental result

Our proposed FA-AIB and FA-AIB-s are compared with the state-of-the-art Fast-AIB in [7] in terms of computational load and classification performance. We use the implementation of Fast-AIB in [18]. The computational load includes the time and memory spent in hierarchically merging n words to two word clusters only. The classification performance is evaluated by both classification error rate and the mean Average Precision (mAP), obtained by classifying the data with the words clustered at each level of the hierarchy. Also, the mutual information loss incurred by each algorithm in the clustering process is compared on both training and test data sets. A linear Support Vector Machines classifier is used, and its regularization parameter is equally optimized for all the three algorithms. The platform for timing is a desktop computer with 3.3GHz Core$^{\text{TM}}$ 2 Duo CPU and 3.2GB memory.

4.1. Data sets

One synthetic and three real data sets are used, where different number (from 1000 to $60,000$) of words (or features) are clustered. They are briefly introduced as follows.

Synthetic data set. It is created to give a systematic comparison of computational load. We fix the number of training samples at 100, with 50 positive and 50 negative samples. Each sample is represented by a histogram whose dimensions gradually increase from 1000 to $100,000$, simulating the use of different number of words. Each bin of the histogram is filled with a value randomly selected from 0 to 99. Each histogram is normalized to have a sum of one.

Caltech-101[6]. It consists 101 object categories and one *Background* category. For each image, local patches are densely sampled at size of 16×16 with a step size of 8, and SIFT [15] is used to describe each patch. A visual vocabulary of 1000 words is created by applying k-means clustering to the extracted SIFT features. An image is then represented as a 1000-D histogram of the number of occurrences of each visual word. The ten largest-sized object categories are selected. Each of them is used as the positive class respectively, while the *Background* category is used as the negative class. In each classification task, the images of the two involved classes are randomly partitioned into ten training and test groups. The results averaged over all the groups of the ten classification tasks are compared.

PASCAL-VOC07 [5]. It consists of 20 object categories and 9963 images in total. The released predefined training and test sets are used. A visual vocabulary of 4000 words is generated in the same way as described in Caltech101. Each SIFT feature in an image is coded by the localized soft-assignment coding [13]. All the coding coefficients are then max-pooled to form a 4000-D feature vector to represent this image. Also, we apply Spatial Pyramid [10] to divide an image into 1×1, 3×1 (three horizontal stripes) and 2×2 (four quadrants) grids, for a total of 8 regions. One 4000-D feature vector is constructed for each region respectively, and this generates for each image a representation of higher dimension of $4000 \times 8 = 32,000$. Each of the 20 object categories is used as the positive class and the remaining 19 ones are used as the negative class. The results averaged over all the 20 tasks are computed for comparison.

20Newsgroups [8]. It consists around $20,000$ documents, nearly evenly distributed over 20 news groups. It is a benchmark data set used for evaluating learning algorithms for text analysis. The version in this experiment includes $15,935$ training documents and 3993 test documents[2]. Each document is represented by a $62,061$-D histogram. Again, each group is discriminated from the remaining groups respectively, leading to 20 binary classification tasks. The results are averaged over the 20 tasks for comparison.

4.2. Comparison of computational efficiency

This experiment verifies on both synthetic and real data sets that our algorithms achieve higher computational efficiency. The result on the synthetic data set is in Figure 3. The x-axis is the number of words to be clustered and the y-axis is the time spent by each algorithm. Also, some detailed timing result is provided in Table 2. As shown, our algorithms consistently improve the speed over Fast-AIB,

[2]http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multiclass.html.

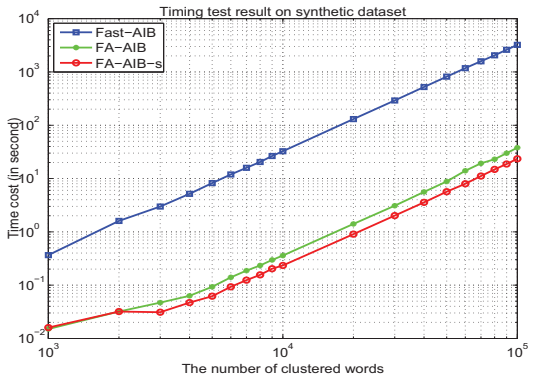

Figure 3. Comparison of running time of three algorithms.

Table 2. Time cost on synthetic dataset (in second)

# of words	Fast-AIB [7]	FA-AIB	FA-AIB-s
1000	0.37	0.016	0.016
5000	8.21	0.093	0.062
10000	32.59	0.36	0.23
20000	130.4	1.41	0.91
50000	817.7	8.79	5.65
100000	3223.8	37.9	23.49

Table 3. Time cost on three real datasets (in second)

Dataset	# of words	Fast-AIB [7]	FA-AIB	FA-AIB-s
Caltech101	1000	0.501	0.015	0.01
VOC07	4000	5.13	0.062	0.049
VOC07	32000	337.41	3.76	2.43
20News-	62061	9050.5	7.67	5.03

and the improvement becomes more pronounced with the increasing number of words. For example, when merging $100,000$ words, our algorithms need no more than *one minute* while Fast-AIB costs almost *one hour*. By average, FA-AIB can be faster than Fast-AIB by 86 times, and FA-AIB-s can be faster by 140 times. Table 3 compares the time cost on three real data sets. The result again verifies the computational efficiency of the proposed algorithms, especially on the large VOC07 and 20Newsgroups data sets. The above comparison indicates the remarkable benefit when clustering a large number of words.

4.3. Comparison of classification performance

We investigate if our algorithms can well maintain classification performance. Figure 4 shows the average result for each real data set. The left column plots the classification error rate. As seen, FA-AIB attains very similar classification performance as Fast-AIB. Moreover, it even shows slightly better performance on Caltech101 and 20Newsgroups when the words are merged into a smaller number of word clusters. The simplified variant, FA-AIB-s, works fairly well considering the coarser approximation used by it and its computational superiority. On most data sets, it does not incur significant performance degradation until the words are clustered to a small number of clusters. These results are confirmed by the mean Average Precision (mAP) in the right column. To gain insight into the clustering process, we also plot (in the logarithm of 10) the accumulative mutual information loss at each level of the hierarchy on both training and test sets in Figure 5. As expected, the loss monotonically increases with the progress of clustering. On Caltech101 and PASCAL-VOC07 (rows $1-3$), the loss incurred by Fast-AIB and the proposed FA-AIB overlaps with each other, indicating the almost identical loss of information. Comparatively, FA-AIB-s incurs a bit more loss as expected. This is consistent with the classification error rate and mAP obtained. On 20Newsgroups, Fast-AIB and FA-AIB also produce similar information loss on the training set, and FA-AIB-s incurs slightly more loss. It is worth mentioning that on 20Newsgroups, FA-AIB consistently produces lower information loss on the test set, which agrees well with its slightly better classification performance shown in Figure 4. Observing that the training sample size in 20Newsgroups is much smaller than its number of words, we attribute this improvement to the mild "regularization" effect introduced by the approximation in the proposed FA-AIB.

5. Conclusion and future work

We propose a fast algorithm for agglomerative information bottleneck to do distributional word clustering. Instead of explicitly computing the mutual information loss incurred by merging each pair of words, our algorithm simply utilizes the class ratio of each word to find good candidate word pairs. Theoretical analysis and experimental study show its computational advantage and the well-maintained classification performance. The future work will systematically extend this idea to multi-class classification and to other information-theoretic methods where finding the minimum mutual information loss is intensively needed.

References

[1] L. D. Baker and A. K. McCallum. Distributional clustering of words for text classification. SIGIR, pages 96–103, New York, NY, USA, 1998. ACM. 1

[2] R. Bekkerman, R. El-Yaniv, Y. Winter, and N. Tishby. On feature distributional clustering for text categorization. In *SIGIR*, pages 146–153, 2001. 1

[3] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, Dec. 1992. 1

[4] I. S. Dhillon, S. Mallela, and R. Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *JMLR*, 3:1265–1287, 2003. 1

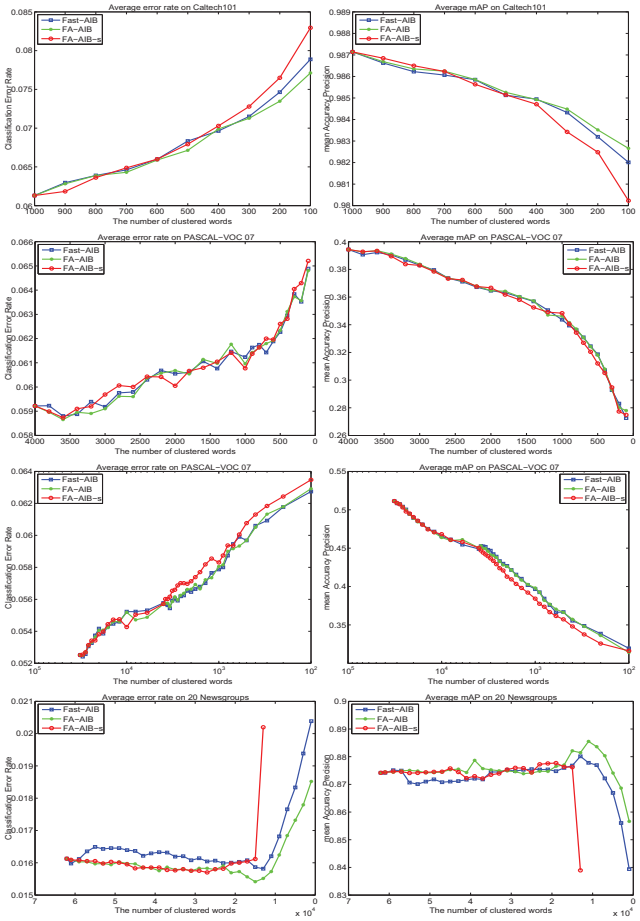

Figure 4. Comparison of classification error rate (left), mean Average Precision (right) on four datasets at each row: Caltech101, VOC07(4000 words), VOC07(32, 000 words), 20Newsgroups.

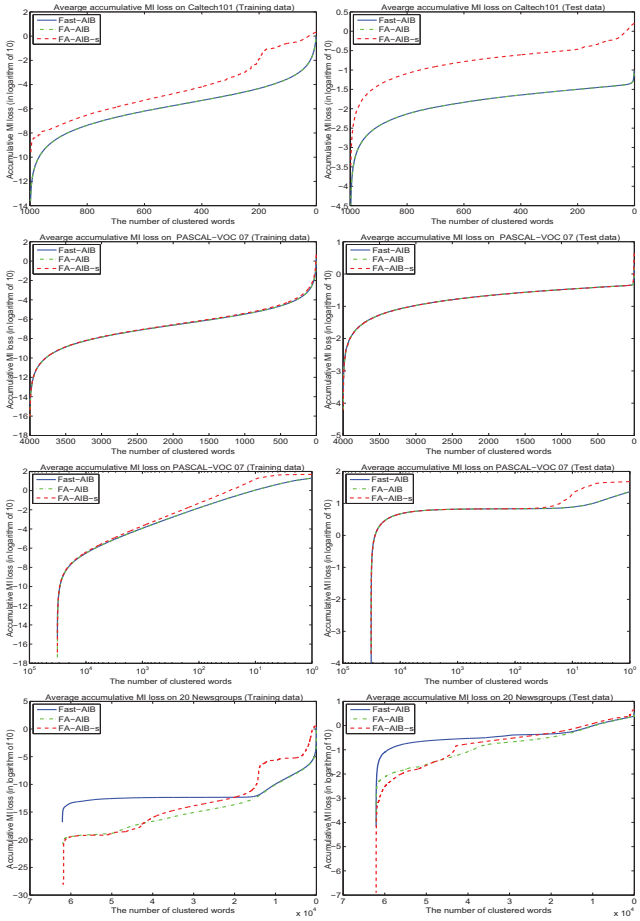

Figure 5. Comparison of mutual information loss on training (left) and test (right) data on four datasets at each row: Caltech101, VOC07(4000 words), VOC07(32, 000 words), 20Newsgroups.

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. 6

[6] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *CVIU*, 106(1):59–70, 2007. 6

[7] B. Fulkerson, A. Vedaldi, and S. Soatto. Localizing objects with smart dictionaries. In *ECCV*, pages 179–192, Berlin, Heidelberg, 2008. Springer-Verlag. 1, 2, 6, 7

[8] K. Lang. Newsweeder: Learning to filter netnews. In *ICML*, pages 331–339, 1995. 6

[9] S. Lazebnik and M. Raginsky. Supervised learning of quantizer codebooks by information loss minimization. *IEEE TPAMI*, 31:1294–1309, 2009. 1

[10] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. CVPR, pages 2169–2178, 2006. 6

[11] J. Liu and M. Shah. Scene modeling using co-clustering. In *ICCV*, pages 1–7, 2007. 1

[12] J. Liu and M. Shah. Learning human actions via information maximization. *CVPR*, 0:1–8, 2008. 1

[13] L. Liu, L. Wang, and X. Liu. In defense of soft-assignment coding. In *ICCV*, pages 2486–2493, 2011. 6

[14] L. Liu, L. Wang, and C. Shen. A generalized probabilistic framework for compact codebook creation. In *CVPR*, pages 1537–1544, 2011. 1

[15] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 6

[16] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of english words. ACL, pages 183–190, Stroudsburg, PA, USA, 1993. Association for Computational Linguistics. 1

[17] N. Slonim and N. Tishby. Agglomerative information bottleneck. In *NIPS*, pages 617–623, 1999. 1, 2

[18] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. Software available at http://www.vlfeat.org/, 2008. 6

[19] L. Wang, L. Zhou, and C. Shen. A fast algorithm for creating a compact and discriminative visual codebook. In *ECCV*, pages 719–732, 2008. 1

[20] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, pages 1800–1807, 2005. 1