Discovering Word Senses from Text

Patrick Pantel and Dekang Lin
University of Alberta
Department of Computing Science
Edmonton, Alberta T6H 2E1 Canada
{ppantel, lindek}@cs.ualberta.ca

ABSTRACT

Inventories of manually compiled dictionaries usually serve as a source for word senses. However, they often include many rare senses while missing corpus/domain-specific senses. We present a clustering algorithm called CBC (Clustering By Committee) that automatically discovers word senses from text. It initially discovers a set of tight clusters called committees that are well scattered in the similarity space. The centroid of the members of a committee is used as the feature vector of the cluster. We proceed by assigning words to their most similar clusters. After assigning an element to a cluster, we remove their overlapping features from the element. This allows CBC to discover the less frequent senses of a word and to avoid discovering duplicate senses. Each cluster that a word belongs to represents one of its senses. We also present an evaluation methodology for automatically measuring the precision and recall of discovered senses.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval---*Clustering*.

General Terms

Algorithms, Measurement, Experimentation.

Keywords

Word sense discovery, clustering, evaluation, machine learning.

1. INTRODUCTION

Using word senses versus word forms is useful in many applications such as information retrieval [20], machine translation [5] and question-answering [16]. In previous approaches, word senses are usually defined using a manually constructed lexicon. There are several disadvantages associated with these word senses. First, manually created lexicons often contain rare senses. For example, WordNet 1.5 [15] (hereon referred to as WordNet) included a sense of *computer* that means 'the person who computes'. Using WordNet to expand queries to an information retrieval system, the expansion of *computer* includes words like *estimator* and *reckoner*. The second problem with these lexicons is that they miss many domain specific senses. For example, WordNet misses the user-interface-object sense of the word *dialog* (as often used in software manuals).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGKDD'02, July 23-26, 2002, Edmonton, Alberta, Canada.
Copyright 2002 ACM 1-58113-567-X/02/0007…$5.00.

The meaning of an unknown word can often be inferred from its context. Consider the following sentences:

```
A bottle of tezgüno is on the table.
Everyone likes tezgüno.
Tezgüno makes you drunk.
We make tezgüno out of corn.
```

The contexts in which the word *tezgüno* is used suggest that *tezgüno* may be a kind of alcoholic beverage. This is because other alcoholic beverages tend to occur in the same contexts as *tezgüno*. The intuition is that words that occur in the same contexts tend to be similar. This is known as the Distributional Hypothesis [3]. There have been many approaches to compute the similarity between words based on their distribution in a corpus [4][8][12]. The output of these programs is a ranked list of similar words to each word. For example, [12] outputs the following similar words for *wine* and *suit*:

```
wine:  beer, white wine, red wine, Chardonnay,
       champagne, fruit, food, coffee, juice,
       Cabernet, cognac, vinegar, Pinot noir,
       milk, vodka,…

suit:  lawsuit, jacket, shirt, pant, dress,
       case, sweater, coat, trouser, claim,
       business suit, blouse, skirt,
       litigation, …
```

The similar words of *wine* represent the meaning of wine. However, the similar words of *suit* represent a mixture of its *clothing* and *litigation* senses. Such lists of similar words do not distinguish between the multiple senses of polysemous words.

The algorithm we present in this paper automatically discovers word senses by clustering words according to their distributional similarity. Each cluster that a word belongs to corresponds to a sense of the word. Consider the following sample outputs from our algorithm:

```
(suit
  Nq34   0.39 (blouse, slack, legging,
         sweater)
  Nq137  0.20 (lawsuit, allegation, case,
         charge)
)
(plant
  Nq215  0.41 (plant, factory, facility,
         refinery)
  Nq235  0.20 (shrub, ground cover,
         perennial, bulb)
)
```

```
(heart
  Nq72   0.27 (kidney, bone marrow, marrow,
         liver)
  Nq866  0.17 (psyche, consciousness, soul,
         mind)
)
```

Each entry shows the clusters to which the headword belongs. Nq34, Nq137, … are automatically generated names for the clusters. The number after each cluster name is the similarity between the cluster and the headword (i.e. *suit*, *plant* and *heart*). The lists of words are the top-4 most similar members to the cluster centroid. Each cluster corresponds to a sense of the headword. For example, Nq34 corresponds to the *clothing* sense of *suit* and Nq137 corresponds to the *litigation* sense of *suit*.

In this paper, we present a clustering algorithm, CBC (Clustering By Committee), in which the centroid of a cluster is constructed by averaging the feature vectors of a subset of the cluster members. The subset is viewed as a committee that determines which other elements belong to the cluster. By carefully choosing committee members, the features of the centroid tend to be the more typical features of the target class.

We also propose an automatic evaluation methodology for senses discovered by clustering algorithms. Using the senses in WordNet, we measure the precision of a system's discovered senses and the recall of the senses it should discover.

2. RELATED WORK

Clustering algorithms are generally categorized as hierarchical and partitional. In hierarchical agglomerative algorithms, clusters are constructed by iteratively merging the most similar clusters. These algorithms differ in how they compute cluster similarity. In single-link clustering, the similarity between two clusters is the similarity between their most similar members while complete-link clustering uses the similarity between their least similar members. Average-link clustering computes this similarity as the average similarity between all pairs of elements across clusters. The complexity of these algorithms is $O(n^2 \log n)$, where n is the number of elements to be clustered [6].

Chameleon is a hierarchical algorithm that employs dynamic modeling to improve clustering quality [7]. When merging two clusters, one might consider the sum of the similarities between pairs of elements across the clusters (e.g. average-link clustering). A drawback of this approach is that the existence of a single pair of very similar elements might unduly cause the merger of two clusters. An alternative considers the number of pairs of elements whose similarity exceeds a certain threshold [3]. However, this may cause undesirable mergers when there are a large number of pairs whose similarities barely exceed the threshold. Chameleon clustering combines the two approaches.

K-means clustering is often used on large data sets since its complexity is linear in n, the number of elements to be clustered. *K*-means is a family of partitional clustering algorithms that iteratively assigns each element to one of K clusters according to the centroid closest to it and recomputes the centroid of each cluster as the average of the cluster's elements. However, *K*-means has complexity $O(K \times T \times n)$ and is efficient for many clustering tasks. Because the initial centroids are randomly selected, the resulting clusters vary in quality. Some sets of initial centroids lead to poor convergence rates or poor cluster quality.

Bisecting *K*-means [19], a variation of *K*-means, begins with a set containing one large cluster consisting of every element and iteratively picks the largest cluster in the set, splits it into two clusters and replaces it by the split clusters. Splitting a cluster consists of applying the basic *K*-means algorithm α times with K=2 and keeping the split that has the highest average element-centroid similarity.

Hybrid clustering algorithms combine hierarchical and partitional algorithms in an attempt to have the high quality of hierarchical algorithms with the efficiency of partitional algorithms. Buckshot [1] addresses the problem of randomly selecting initial centroids in *K*-means by combining it with average-link clustering. Cutting et al. claim its clusters are comparable in quality to hierarchical algorithms but with a lower complexity. Buckshot first applies average-link to a random sample of $\sqrt{n}$ elements to generate K clusters. It then uses the centroids of the clusters as the initial K centroids of *K*-means clustering. The sample size counterbalances the quadratic running time of average-link to make Buckshot efficient: $O(K \times T \times n + n \log n)$. The parameters K and T are usually considered to be small numbers.

CBC is a descendent of UNICON [13], which also uses small and tight clusters to construct initial centroids. We compare them in Section 4.4 after presenting the CBC algorithm.

3. WORD SIMILARITY

Following [12], we represent each word by a feature vector. Each feature corresponds to a context in which the word occurs. For example, "sip __" is a verb-object context. If the word *wine* occurred in this context, the context is a feature of *wine*. The value of the feature is the pointwise mutual information [14] between the feature and the word. Let c be a context and $F_c(w)$ be the frequency count of a word w occurring in context c. The pointwise mutual information, $mi_{w,c}$, between c and w is defined as:

$$mi_{w,c} = \frac{\frac{F_c(w)}{N}}{\frac{\sum_i F_i(w)}{N} \times \frac{\sum_j F_c(j)}{N}} \quad \textbf{(1)}$$

where $N = \sum_i \sum_j F_i(j)$ is the total frequency counts of all words and their contexts. A well-known problem with mutual information is that it is biased towards infrequent words/features. We therefore multiplied $mi_{w,c}$ with a discounting factor:

$$\frac{F_c(w)}{F_c(w)+1} \times \frac{min\left(\sum_i F_i(w), \sum_j F_c(j)\right)}{min\left(\sum_i F_i(w), \sum_j F_c(j)\right)+1} \quad \textbf{(2)}$$

We compute the similarity between two words w_i and w_j using the *cosine coefficient* [17] of their mutual information vectors:

$$sim(w_i, w_j) = \frac{\sum_c mi_{w_i c} \times mi_{w_j c}}{\sqrt{\sum_c mi_{w_i c}^{\ 2} \times \sum_c mi_{w_j c}^{\ 2}}} \quad \textbf{(3)}$$

4. ALGORITHM

CBC consists of three phases. In Phase I, we compute each element's top-k similar elements. In our experiments, we used k = 10. In Phase II, we construct a collection of tight clusters, where the elements of each cluster form a **committee**. The algorithm tries to form as many committees as possible on the condition that each newly formed committee is not very similar to any existing committee. If the condition is violated, the committee is simply discarded. In the final phase of the algorithm, each element e is assigned to its most similar clusters.

4.1 Phase I: Find top-similar elements

Computing the complete similarity matrix between pairs of elements is obviously quadratic. However, one can dramatically reduce the running time by taking advantage of the fact that the feature vector is sparse. By indexing the features, one can retrieve the set of elements that have a given feature. To compute the top similar elements of an element e, we first sort the features according to their pointwise mutual information values and then only consider a subset of the features with highest mutual information. Finally, we compute the pairwise similarity between e and the elements that share a feature from this subset. Since high mutual information features tend not to occur in many elements, we only need to compute a fraction of the possible pairwise combinations. Using this heuristic, similar words that share only low mutual information features will be missed by our algorithm. However, in our experiments, this had no visible impact on cluster quality.

4.2 Phase II: Find committees

The second phase of the clustering algorithm recursively finds tight clusters scattered in the similarity space. In each recursive step, the algorithm finds a set of tight clusters, called committees, and identifies residue elements that are not covered by any committee. We say a committee **covers** an element if the element's similarity to the centroid of the committee exceeds some high similarity threshold. The algorithm then recursively attempts to find more committees among the residue elements. The output of the algorithm is the union of all committees found in each recursive step. The details of Phase II are presented in Figure 1.

In Step 1, the score reflects a preference for bigger and tighter clusters. Step 2 gives preference to higher quality clusters in Step 3, where a cluster is only kept if its similarity to all previously kept clusters is below a fixed threshold. In our experiments, we set $\theta_1 = 0.35$. Step 4 terminates the recursion if no committee is found in the previous step. The residue elements are identified in Step 5 and if no residues are found, the algorithm terminates; otherwise, we recursively apply the algorithm to the residue elements.

Each committee that is discovered in this phase defines one of the final output clusters of the algorithm.

4.3 Phase III: Assign elements to clusters

In Phase III, each element e is assigned to its most similar clusters in the following way:

```
let C be a list of clusters initially empty
let S be the top-200 similar clusters to e
```

Input: A list of elements E to be clustered, a similarity database S from Phase I, thresholds θ_1 and θ_2.

Step 1: For each element $e \in E$

Cluster the top similar elements of e from S using average-link clustering.

For each cluster discovered c compute the following score: $|c| \times \text{avgsim}(c)$, where $|c|$ is the number of elements in c and avgsim(c) is the average pairwise similarity between elements in c.

Store the highest-scoring cluster in a list L.

Step 2: Sort the clusters in L in descending order of their scores.

Step 3: Let C be a list of committees, initially empty.

For each cluster $c \in L$ in sorted order

Compute the centroid of c by averaging the frequency vectors of its elements and computing the mutual information vector of the centroid in the same way as we did for individual elements.

If c's similarity to the centroid of each committee previously added to C is below a threshold θ_1, add c to C.

Step 4: If C is empty, we are done and return C.

Step 5: For each element $e \in E$

If e's similarity to every committee in C is below threshold θ_2, add e to a list of residues R.

Step 6: If R is empty, we are done and return C.

Otherwise, return the union of C and the output of a recursive call to Phase II using the same input except replacing E with R.

Output: a list of committees.

Figure 1. Phase II of CBC.

```
while S is not empty {
  let c∈S be the most similar cluster to e
  if the similarity(e, c) < σ
    exit the loop
  if c is not similar to any cluster in C {
    assign e to c
    remove from e its features that overlap
      with the features of c;
  }
  remove c from S
}
```

When computing the similarity between a cluster and an element (or another cluster) we use the centroid of committee members as the representation for the cluster. This phase resembles K-means in that elements are assigned to their closest centroids. Unlike K-means, the number of clusters is not fixed and the centroids do not change (i.e. when an element is added to a cluster, it is not added to the committee of the cluster).

The key to the algorithm for discovering senses is that once an element e is assigned to a cluster c, the intersecting features

between *e* and *c* are removed from *e*. This allows CBC to discover the less frequent senses of a word and to avoid discovering duplicate senses.

4.4 Comparison with UNICON

UNICON [13] also constructs cluster centroids using a small set of similar elements, like the committees in CBC.

One of the main differences between UNICON and CBC is that UNICON only guarantees that the committees do not have overlapping members. However, the centroids of two committees may still be quite similar. UNICON deals with this problem by merging such clusters. In contrast, Step 2 in Phase II of CBC only outputs a committee if its centroid is not similar to any previously output committee.

Another main difference between UNICON and CBC is in Phase III of CBC. UNICON has difficulty discovering senses of a word when this word has a dominating sense. For example, in the newspaper corpus that we used in our experiments, the *factory* sense of *plant* is used much more frequently than its *life* sense. Consequently, the majority of the features of the word *plant* are related to its *factory* sense. This is evidenced in the following top-30 most similar words of *plant*.

```
facility, factory, reactor, refinery, power
plant, site, manufacturing plant, tree,
building, complex, landfill, dump, project,
mill, airport, station, farm, operation,
warehouse, company, home, center, lab, store,
industry, park, house, business, incinerator
```

All of the above, except the word *tree*, are related to the *factory* sense. Even though UNICON generated a cluster

```
ground cover, perennial, shrub, bulb, annual,
wildflower, shrubbery, fern, grass, ...
```

the similarity between *plant* and this cluster is very low.

On the other hand, CBC removes the *factory* related features from the feature vector of *plant* after it is assigned to the *factory* cluster. As a result, the similarity between the {`ground cover, perennial, …`} cluster and the revised feature vector of *plant* becomes much higher.

5. EVALUATION METHODOLOGY

To evaluate our system, we compare its output with WordNet, a manually created lexicon.

5.1 WordNet

WordNet [15] is an electronic dictionary organized as a graph. Each node, called a **synset**, represents a set of synonymous words. The arcs between synsets represent **hyponym/hypernym** (subclass/superclass) relationships[1]. Figure 2 shows a fragment of WordNet. The number attached to a synset *s* is the probability that a randomly selected noun refers to an instance of *s* or any synset below it. These probabilities are not included in WordNet. We use the frequency counts of synsets in the SemCor [9] corpus to estimate them. Since SemCor is a fairly small corpus (200K words), the frequency counts of the synsets in the lower part of the WordNet hierarchy are very sparse. We smooth the probabilities by assuming that all siblings are equally likely given the parent.

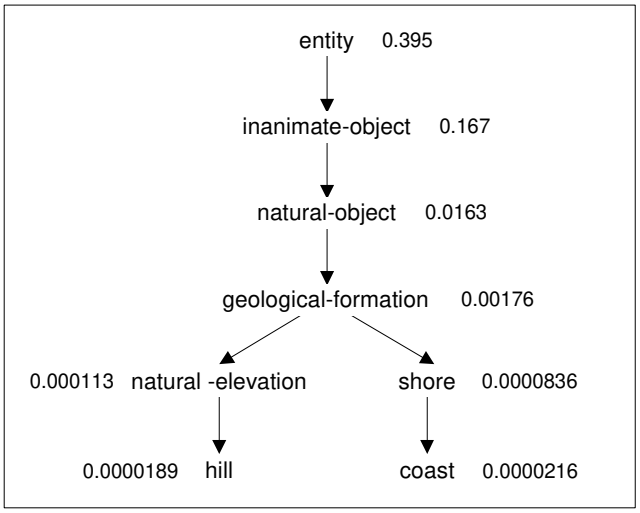

Figure 2. Example hierarchy of synsets in WordNet along with each synset's probability.

Lin [11] defined the similarity between two WordNet synsets s_1 and s_2 as:

$$sim(s_1, s_2) = \frac{2 \times log\ P(s)}{log\ P(s_1) + log\ P(s_2)} \quad \textbf{(4)}$$

where *s* is the most specific synset that subsumes s_1 and s_2. For example, using Figure 2, if s_1 = *hill* and s_2 = *shore* then *s* = *geological-formation* and *sim*(*hill*, *shore*) = 0.626.

5.2 Precision

For each word, CBC outputs a list of clusters to which the word belongs. Each cluster should correspond to a sense of the word. The **precision** of the system is measured by the percentage of output clusters that actually correspond to a sense of the word.

To compute the precision, we must define what it means for a cluster to correspond to a correct sense of a word. To determine this automatically, we map clusters to WordNet senses.

Let $\boldsymbol{S(w)}$ be the set of WordNet senses of a word *w* (each sense is a synset that contains *w*). We define *simW*(*s*, *u*), the similarity between a synset *s* and a word *u*, as the maximum similarity between *s* and a sense of *u*:

$$simW(s,u) = \max_{t \in S(u)} sim(s,t) \quad \textbf{(5)}$$

Let $\boldsymbol{c_k}$ be the top-*k* members of a cluster *c*, where these are the *k* most similar members to the committee of *c*. We define the similarity between *s* and *c*, *simC*(*s*, *c*), as the average similarity between *s* and the top-*k* members of *c*:

[1] WordNet also contains other semantic relationships such as meronyms (part-whole relationships) and antonyms, however we do not use them here.

$$simC(s,c) = \frac{\sum_{u \in c_k} simW(s,u)}{k} \quad (6)$$

Suppose a clustering algorithm assigns the word w to cluster c. We say that c corresponds to a **correct sense** of w if

$$\max_{s \in S(w)} simC(s,c) \geq \theta \quad (7)$$

In our experiments, we set k = 4 and varied the θ values. The WordNet sense of w that corresponds to c is then:

$$\arg\max_{s \in S(w)} simC(s,c) \quad (8)$$

It is possible that multiple clusters will correspond to the same WordNet sense. In this case, we only count one of them as correct.

We define the **precision** of a word w as the percentage of correct clusters to which it is assigned. The precision of a clustering algorithm is the average precision of all the words.

5.3 Recall

The **recall** (completeness) of a word w measures the ratio between the correct clusters to which w is assigned and the actual number of senses in which w was used in the corpus. Clearly, there is no way to know the complete list of senses of a word in any non-trivial corpus. To address this problem, we pool the results of several clustering algorithms to construct the target senses. For a given word w, we use the union of the correct cluster of w discovered by the algorithms as the target list of senses for w.

While this recall value is likely not the true recall, it does provide a relative ranking of the algorithms used to construct the pool of target senses. The overall recall is the average recall of all words.

5.4 *F*-measure

The *F*-measure [18] combines precision and recall aspects:

$$F = \frac{2RP}{R+P} \quad (9)$$

where R is the recall and P is the precision. F weights low values of precision and recall more heavily than higher values. It is high when both precision and recall are high.

6. EXPERIMENTAL RESULTS

In this section, we describe our experimental setup and present evaluation results of our system.

6.1 Setup

We used Minipar[2] [10], a broad-coverage English parser, to parse about 1GB (144M words) of newspaper text from the TREC collection (1988 AP Newswire, 1989-90 LA Times, and 1991 San Jose Mercury) at a speed of about 500 words/second on a PIII-750 with 512MB memory. We collected the frequency counts of the grammatical relationships (contexts) output by Minipar and used them to compute the pointwise mutual information values from Section 3. The test set is constructed by intersecting the words in WordNet with the nouns in the corpus whose total mutual information with all of its contexts exceeds a threshold (we used 250). Since WordNet has a low coverage of proper names, we removed all capitalized nouns. The resulting test set consists of 13403 words. The average number of features per word is 740.8.

[2]Available at www.cs.ualberta.ca/~lindek/minipar.htm.

Table 1. Precision, Recall and *F*-measure on the data set for various algorithms with σ = 0.18 and θ = 0.25.

Algorithm	*Precision (%)*	*Recall (%)*	*F-measure (%)*
CBC	**60.8**	**50.8**	**55.4**
UNICON	53.3	45.5	49.2
Buckshot	52.6	45.2	48.6
K-means	48.0	44.2	46.0
Bisecting *K*-means	33.8	31.8	32.8
Average-link	50.0	41.0	45.0

We modified the average-link, *K*-means, Bisecting *K*-means and Buckshot algorithms of Section 2 since these algorithms only assign each element to a single cluster. For each of these algorithms, the modification is as follows:

```
Apply the algorithm as described in Section 2
For each cluster c returned by the algorithm
  Create a centroid for c using all elements
    assigned to it
Apply MK-means using the above centroids
```

where *MK*-means is the *K*-means algorithm, using the above centroids as initial centroids, except that each element is assigned to its most similar cluster plus all other clusters with which it has similarity greater than σ. We then use these modified algorithms to discover senses.

These clustering algorithms were not designed for sense discovery. Like UNICON, when assigning an element to a cluster, they do not remove the overlapping features from the element. Thus, a word is often assigned to multiple clusters that are similar.

6.2 Word Sense Evaluation

We ran CBC and the modified clustering algorithms described in the previous subsection on the data set and applied the evaluation methodology from Section 4.4. Table 1 shows the results. For Buckshot and *K*-means, we set the number of clusters to 1000 and the maximum number of iterations to 5. For the Bisecting *K*-means algorithm, we applied the basic *K*-means algorithm twice (α = 2 in Section 2) with a maximum of 5 iterations per split. CBC returned 941 clusters and outperformed the next best algorithm by 7.5% on precision and 5.3% on recall.

In Section 5.2 we stated that a cluster corresponds to a correct sense of a word w if its maximum *simC* similarity with any synset

Table 2. Comparison of manual and automatic evaluations of a 1% random sample of the data set.

		MANUAL		
		√	×	+
AUTOMATIC	√	**104**	2	0
	×	17	**41**	0
	+	0	1	**3**

in $S(w)$ exceeds a threshold θ (Eq. 7). Figure 2 shows our experiments using different values of θ. The higher the θ value, the stricter we are in defining correct senses. Naturally, the systems' *F*-measures decrease when θ increases. The relative ranking of the algorithms is not sensitive to the choice of θ values. CBC has higher *F*-measure for all θ thresholds.

For all sense discovery algorithms, we assign an element to a cluster if their similarity exceeds a threshold σ. The value of σ does not affect the first sense returned by the algorithms for each word because each word is always assigned to its most similar cluster. We experimented with different values of σ and present the results in Figure 3. With a lower σ value, words are assigned to more clusters. Consequently, the precision goes down while recall goes up. CBC has higher *F*-measure for all σ thresholds.

6.3 Manual Evaluation

We manually evaluated a 1% random sample of the test data consisting of 133 words with 168 senses. Here is a sample of the instances that are manually judged for the words *aria*, *capital* and *device*:

```
aria     S1: song, ballad, folk song, tune
capital  S1: money, donation, funding,
             honorarium
capital  S2: camp, shantytown, township, slum
device   S1: camera, transmitter, sensor,
             electronic device
device   S2: equipment, test equipment,
             microcomputer, video equipment
```

For each discovered sense of a word, we include its top-4 most similar words. The evaluation consists of assigning a tag to each sense as follows:

- √: The list of top-4 words describes a sense of the word that has not yet been seen
- +: The list of top-4 words describes a sense of the word that has already been seen (duplicate sense)
- ×: The list of top-4 words does not describe a sense of the word

The S2 sense of *device* is an example of a sense that is evaluated with the duplicate sense tag.

Table 2 compares the agreements/disagreements between our manual and automatic evaluations. Our manual evaluation agreed with the automatic evaluation 88.1% of the time. This suggests that the evaluation methodology is reliable.

Most of the disagreements (17 out of 20) were on senses that were incorrect according to the automatic evaluation but correct in the manual evaluation. The automatic evaluation misclassified these because sometimes WordNet misses a sense of a word and because of the organization of the WordNet hierarchy. Some words in WordNet should have high similarity (e.g. *elected official* and *legislator*) but they are not close to each other in the hierarchy.

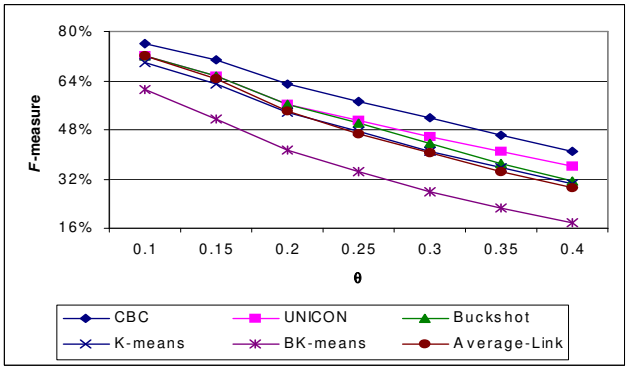

Figure 2. *F*-measure of several algorithms with σ = 0.18 and varying θ thresholds from Eq.7.

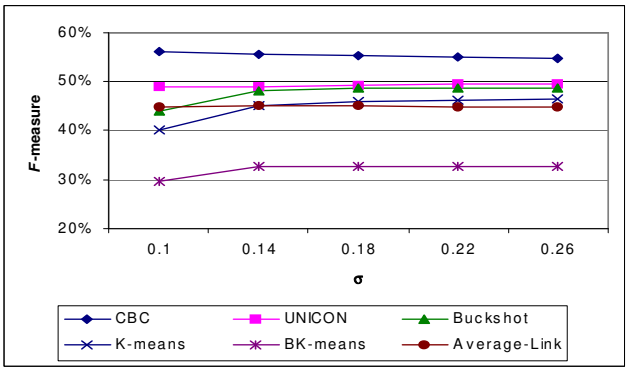

Figure 3. *F*-measure of several algorithms with θ = 0.25 and varying σ thresholds.

Our manual evaluation of the sample gave a precision of 72.0%. The automatic evaluation of the same sample gave 63.1% precision. Of the 13,403 words in the test data, CBC found 2869 of them polysemous.

7. DISCUSSION

We computed the average precision for each cluster, which is the percentage of elements in a cluster that correctly correspond to a WordNet sense according to Eq.7. We inspected the low-precision clusters and found that they were low for three main reasons.

First, some clusters suffer from part-of-speech confusion. Many of the nouns in our data set can also be used as verbs and adjectives. Since the feature vector of a word is constructed from all instances of that word (including its noun, verb and adjective usage), CBC outputs contain clusters of verbs and adjectives. For example, the following cluster contains 112 adjectives:

```
weird, stupid, silly, old, bad, simple,
normal, wrong, wild, good, romantic, tough,
special, small, real, smart, ...
```

The noun senses of all of these words in WordNet are not similar. Therefore, the cluster has a very low 2.6% precision. In hindsight, we should have removed the verb and adjective usage features.

Secondly, CBC outputs some clusters of proper names. If a word that first occurs as a common noun also has a proper-noun usage it will not be removed from the test data. For the same reasons as the part-of-speech confusion problem, CBC discovers proper name clusters but gets them evaluated as if they were common nouns (since WordNet contains few proper nouns). For example, the following cluster has an average precision of 10%:

```
blue jay, expo, angel, mariner, cub, brave,
pirate, twin, athletics, brewer
```

Finally, some concepts discovered by CBC are completely missing from WordNet. For example, the following cluster of government departments has a low precision of 3.3% because WordNet does not have a synset that subsumes these words:

```
public works, city planning, forestry,
finance, tourism, agriculture, health,
affair, social welfare, transport, labor,
communication, environment, immigration,
public service, transportation, urban
planning, fishery, aviation,
telecommunication, mental health,
procurement, intelligence, custom, higher
education, recreation, preservation, lottery,
correction, scouting
```

Somewhat surprisingly, all of the low-precision clusters that we inspected are reasonably good. At first sight, we thought the following cluster was bad:

```
shamrock, nestle, dart, partnership, haft,
consortium, blockbuster, whirlpool, delta,
hallmark, rosewood, odyssey, bass, forte,
cascade, citadel, metropolitan, hooker
```

By looking at the features of the centroid of this cluster, we realized that it is mostly a cluster of company names.

8. CONCLUSION

We presented a clustering algorithm, CBC, that automatically discovers word senses from text. We first find well-scattered tight clusters called committees and use them to construct the centroids of the final clusters. We proceed by assigning words to their most similar clusters. After assigning an element to a cluster, we remove their overlapping features from the element. This allows CBC to discover the less frequent senses of a word and to avoid discovering duplicate senses. Each cluster that a word belongs to represents one of its senses.

We also presented an evaluation methodology for automatically measuring the precision and recall of discovered senses. In our experiments, we showed that CBC outperforms several well known hierarchical, partitional, and hybrid clustering algorithms. Our manual evaluation of sample CBC outputs agreed with 88.1% of the decisions made by the automatic evaluation.

9. ACKNOWLEDGEMENTS

The authors wish to thank the reviewers for their helpful comments. This research was partly supported by Natural Sciences and Engineering Research Council of Canada grant OGP121338 and scholarship PGSB207797.

10. REFERENCES

[1] Cutting, D. R.; Karger, D.; Pedersen, J.; and Tukey, J. W. 1992. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of SIGIR-92*. pp. 318–329. Copenhagen, Denmark.

[2] Guha, S.; Rastogi, R.; and Kyuseok, S. 1999. ROCK: A robust clustering algorithm for categorical attributes. In *Proceedings of ICDE'99*. pp. 512–521. Sydney, Australia.

[3] Harris, Z. 1985. Distributional structure. In: Katz, J. J. (ed.) *The Philosophy of Linguistics*. New York: Oxford University Press. pp. 26–47.

[4] Hindle, D. 1990. Noun classification from predicate-argument structures. In *Proceedings of ACL-90*. pp. 268–275. Pittsburgh, PA.

[5] Hutchins, J. and Sommers, H. 1992. *Introduction to Machine Translation*. Academic Press.

[6] Jain, A. K.; Murty, M. N.; and Flynn, P. J. 1999. Data clustering: A review. ACM Computing Surveys 31(3):264–323.

[7] Karypis, G.; Han, E.-H.; and Kumar, V. 1999. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer: Special Issue on Data Analysis and Mining* 32(8):68–75.

[8] Landauer, T. K., and Dumais, S. T. 1997. A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review* 104:211–240.

[9] Landes, S.; Leacock, C.; and Tengi, R. I. 1998. Building semantic concordances. In *WordNet: An Electronic Lexical Database*, edited by C. Fellbaum. pp. 199–216. MIT Press.

[10] Lin, D. 1994. Principar - an efficient, broad-coverage, principle-based parser. *Proceedings of COLING-94*. pp. 42–48. Kyoto, Japan.

[11] Lin, D. 1997. Using syntactic dependency as local context to resolve word sense ambiguity. In *Proceedings of ACL-97*. pp. 64–71. Madrid, Spain.

[12] Lin, D. 1998. Automatic retrieval and clustering of similar words. *Proceedings of COLING/ACL-98*. pp. 768–774. Montreal, Canada.

[13] Lin, D. and Pantel, P. 2001. Induction of semantic classes from natural language text. In *Proceedings of SIGKDD-01*. pp. 317–322. San Francisco, CA.

[14] Manning, C. D. and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

[15] Miller, G. 1990. WordNet: An online lexical database. *International Journal of Lexicography*, 1990.

[16] Pasca, M. and Harabagiu, S. 2001. The informative role of WordNet in Open-Domain Question Answering. In *Proceedings of NAACL-01 Workshop on WordNet and Other Lexical Resources*. pp. 138–143. Pittsburgh, PA.

[17] Salton, G. and McGill, M. J. 1983. *Introduction to Modern Information Retrieval*. McGraw Hill.

[18] Shaw Jr, W. M.; Burgin, R.; and Howell, P. 1997. Performance standards and evaluations in IR test collections: Cluster-based retrieval methods. *Information Processing and Management* 33:1–14, 1997.

[19] Steinbach, M.; Karypis, G.; and Kumar, V. 2000. A comparison of document clustering techniques. *Technical Report #00-034*. Department of Computer Science and Engineering, University of Minnesota.

[20] Voorhees, E. M. 1998. Using WordNet for text retrieval. In *WordNet: An Electronic Lexical Database*, edited by C. Fellbaum. pp. 285–303. MIT Press.