Clustering Using Feature Domain Similarity to Discover Word Senses for Adjectives

Noriko Tomuro, Steven L. Lytinen
DePaul University,
School of Computer Science, Telecommunications and Information Systems,
Chicago, IL 60604 USA
{tomuro,lytinen}@cs.depaul.edu

Kyoko Kanzaki, Hitoshi Isahara
National Institute of Information and Communications Technology,
Soraku-gun, Kyoto, 619-0289 Japan
{kanzaki,isahara}@nict.go.jp

Abstract

This paper presents a new clustering algorithm called DSCBC *which is designed to automatically discover word senses for polysemous words. DSCBC is an extension of CBC Clustering [11], and incorporates* feature domain similarity*: the similarity between the features* themselves*, obtained a priori from sources external to the dataset used at hand. When polysemous words are clustered, words that have similar sense patterns are often grouped together, producing* polysemous clusters*: a cluster in which features in several different domains are mixed in. By incorporating the feature domain similarity in clustering, DSCBC produces monosemous clusters, thereby discovering individual senses of polysemous words. In this work, we apply the algorithm to English adjectives, and compare the discovered senses against WordNet. The results show significant improvements by our algorithm over other clustering algorithms including CBC.*

1. Introduction

In Natural Language Processing (NLP), automatic construction of thesauri from corpus data has been an active topic of research. Most conventional thesauri are manually compiled by lexicographers, therefore often contain inconsistencies. Word senses are also sometimes too fine-grained or uncommon, as frequently pointed out for WordNet [10]. A better approach is to automatically derive word meanings from real data.

In the NLP research, many techniques have been developed which discover word senses or semantic classes from corpora (e.g. [4, 9]). Most of them use distributional similarity (the similarity of the contexts in which words occurred) to group words and derive clusters of similar words (i.e. synonyms). However, most of these techniques focused on nouns, which are largely monosemous, so their applicability to polysemous words such as verbs and adjectives is unknown. For relatively polysemous parts of speech, Walde [18] and Boleda et al. [1] present clustering experiments which automatically derive semantic classes for German verbs and Catalan adjectives respectively. However, they both use standard clustering methods (e.g. K-means) only, and no attempt is made to develop new techniques tailored to polysemous words.

Pantel and Lin [11] develop a clustering algorithm called *CBC* (Clustering By Committee), and report a superior performance of CBC over other clustering algorithms in discovering word senses from corpus texts. However, their results are shown again mostly for nouns, and in fact, the algorithm does not include any special considerations to avoid generating *polysemous clusters*: a cluster in which several word meanings are mixed in. The problem of polysemous clusters also arises in other algorithms when the data consists largely of polysemous words. Indeed, some of the semantic classes derived in [18] are polysemous clusters.

In this paper, we present a new clustering algorithm called *DSCBC* which is specifically designed for discovering senses of polysemous words (but works for monosemous words as well). DSCBC is an extension of CBC, and incorporates *feature domain similarity*: a prior knowledge about the similarity between features *themselves*. For example, if data instances x and y are represented by features $f1$ and $f2$, the feature domain similarity refers to the similarity between $f1$ and $f2$ (not between x and y). Feature domain similarity is obtained a priori from sources external to the dataset(s) used at hand. By incorporating the feature domain similarity in clustering, we are able to group data instances which have features in the same or similar *domain*, thereby producing monosemous clusters.

To test the effectiveness of DSCBC, we apply the algo-

rithm to English adjectives. In this work, we use *abstract nouns* as the features to represent adjectives. From a large corpus, we collect many instances where adjectives modify abstract nouns (e.g. "happy feeling") and construct a dataset. Then after applying the algorithm, we compare the automatically derived word senses against those encoded WordNet as the gold standards. The results show significant improvements by DSCBC over other clustering algorithms in discovering polysemous adjective senses.

2. Polysemy of Adjectives

Adjectives are notoriously polysemous. The function of adjectives is to modify or elaborate the meanings of nouns. By being modified by adjectives, nouns will come to bear specific values for their attributes. For example, "warm soup" is (a bowl of) soup which has the value "warm" (moderately hot) for its temperature attribute. Conversely, adjectives take on different meanings, or change its meaning, depending on the nouns they modify as well. For example, when "warm" modifies the noun "person", the meaning is a psychological one, elaborating the personality or the way a person deals with others.

In linguistics, there is a large body of work on adjectives (e.g. [8]), although the attention they have received is much less than that for nouns and verbs. Likewise in NLP, there exist little other works which tackled adjectives specifically. Most of the recent work on adjectives in NLP has focused either on specific applications (e.g. classifying documents according to semantic orientation or subjectivity [19]), or on specific types of adjectives (e.g. event adjectives [7] and gradable adjectives [3]).

Traditionally, meanings of a polysemous word are enumerated in dictionaries. For example, WordNet lists ten senses for "warm". Distinguishing different senses of polysemous words is a difficult problem, especially for adjectives which are often used metaphorically (e.g. "dark conversation" – a discussion on grim topics, or to talk pessimistically). However, salient meanings are distinguishable based on the *domains* of the co-occurring nouns [6].

In [12], Pustejovsky proposes a generative view of adjectival meanings: meanings of adjectives are indifferentiable, and a specific meaning is generated (by coercion) when an adjective is combined with a noun in the context. He also argues that certain combinations of generated meanings, or patterns of meaning shift, are predictable (which he calls *systematic polysemy*). We do generally agree with his view. Our work in this paper can be considered an effort to acquire such specific meanings that arise in the context as abstract *semantic classes* of adjectives, automatically from corpora. Then we hope to go on to investigate the systematic polysemy of adjectives using those classes in our future work.

3. Dataset

In this work, we use *abstract nouns* as the features to represent adjectives, such as "gentle personality" (as versus concrete nouns such as "gentle goats"). Previous studies in linguistics have shown that, when an abstract noun is modified by an adjective, it often correlates with the semantic class of the adjective (e.g. "gentle" as a personality adjective) [17]. Schmit [15] called such abstract nouns *shell nouns*: a class of abstract nouns which function as conceptual shells that are elaborated by other words or clauses in a text. Thus, abstract nouns provide the meanings of adjectives literally and immediately, while with concrete nouns, the meanings must be inferred.

To create a dataset, we used the Web n-gram corpus created by Google Inc. [2]. This corpus contains English word n-grams (uni-grams to five-grams) and their frequency counts obtained from publicly accessible Web pages containing approximately 1 trillion word tokens (collected in January 2006). To extract adjectives co-occurring with abstract nouns, we first selected 277 abstract nouns which we thought represent a broad range of semantic domains (such as feeling, attribute, scale, sensation). Then we extracted instances of the forms 'Adj AbN' and 'AbN BE Adj' (where BE stands for a copula verb, for instance "nature is gentle") from the bi-grams and tri-grams respectively. There were over 15000 adjectives which co-occurred with the selected abstract nouns in the corpus. Then we randomly selected 1500 of such instances to create the dataset.

In the dataset, we represented each adjective by a feature vector, where a feature was an abstract noun co-occurring with the adjective. The value was the *Mutual Information (MI)* computed from the frequency counts in the corpus.[1] The MI between two words x and y, denoted $I(x, y)$, is:

$$I(x, y) = log \frac{p(x, y)}{p(x)p(y)}$$

where $p(z)$ is the probability of a word z, and $p(x, y)$ is the joint probability of x co-occurring with y. MI indicates the mutual dependence between two random variables, where $I(x, y) = 0$ if x and y are independent, or non-zero (actually positive) otherwise. In our case, $I(x, y)$ essentially indicates how well a feature noun predicts (or is correlated with) a given adjective. MI has been often used in NLP as a way to put weights on feature values (e.g. [4]).

4. Sense Discovery Algorithm for Polysemous Words

Our sense discovery algorithm is an extension of *Clustering By Committee (CBC) Clustering* [11]. We first in-

[1] Actually, we took the log of base 10 of the frequency counts, since the values were quite high and their range was wide as well.

Phase I: Find clusters.
Let L be a list of clusters, initially empty.
For each word $w \in W$ in the dataset,
Select a set c of at most n words from W which are the most similar to w, and add c to L if it is not already in L.
Sort L in the descending order of the scores of the clusters.$^{(1)}$

Phase II: Find committees.
Let C be a list of committees, initially empty.
For each cluster $c \in L$ (in the sorted order),
Compute the centroid of c.$^{(2)}$
If it is not similar to any other committee in C, add c to C.

Phase III: Assign words to committees.
For each word $w \in W$,
Select all committees in C whose centroids are similar to w above a threshold θ.
As a committee is selected, remove all of its features from w.

Figure 1. The CBC Algorithm

troduce CBC and discuss its limitations, then describe the modifications we made to CBC to derive our algorithm.

4.1. CBC Algorithm

CBC is an unsupervised clustering algorithm which automatically derives a set of *committees*. A committee is a cluster of words which are very similar to each other – similar in notion to a synonym set (synset). As with synset, each committee corresponds to a word sense. CBC committees are automatically derived by first finding a *tight* cluster of words which are similar to a given word, for every word in the dataset, then selecting a subset of the derived clusters that are dissimilar/orthogonal to each other as committees. After obtaining the committees, CBC discovers the senses of a word by assigning the word to its most similar committees – all committees with which the similarity is above a threshold, and removing the features of the selected committees from the word as they are selected.

Figure 1 shows the overall steps of the CBC algorithm. A cluster found in Phase I is tight in that it consists of at most n similar words (where $n = 10$ is used in [11]). Then, clusters are selected into the set of committees in the descending order of their scores (indicated with (1) in the algorithm). This way, CBC guarantees that only the tight clusters which are also well scattered in the feature space are selected as committees. The score of a cluster c is computed as $|c| \times$ avgsim(c), where $|c|$ is the number of words in c, and avgsim(c) is the average pairwise similarity between the words in c. Note that throughout the CBC algorithm, the cosine coefficient [14] is used to measure the similarity between vectors, and MI is used to compute the values in the word vectors.

However, one problem with CBC is that it does not work well for polysemous words despite its claimed utility. The reason is because the algorithm uses the centroid to represent a committee (indicated with (2) in the algorithm). Consider the adjectives "warm" and "cold". These words have similar sense patterns, in particular, they both have the senses of temperature and personality. Many other adjectives which elaborate temperature such as "cool" have those two meanings as well. Since their word vectors are similar, they are most likely grouped in the same cluster. And if this cluster is selected as a committee (where the algorithm has no mechanisms besides cosine threshold to prevent it), the committee centroid will end up having both senses, thus producing a polysemous cluster. Consequently, when a word is assigned to committees, the most similar committee would naturally be the one in which the word was the member in the first place, thereby failing to discover the two individual senses.

Note that the same problem also exists in other clustering algorithms as well. When polysemous words are clustered, those which have similar polysemous patterns are grouped together, resulting in a heterogeneous, polysemous cluster in which different senses are mixed.

4.2. Feature Domain Similarity

To overcome the polysemous cluster problem discussed above, we developed a new metric which measures the similarity between the features themselves, which we call the *Feature Domain Similarity*. This is the primary characteristics and the contribution of our algorithm.

Consider the following data:

```
   a b c d
x: 1 1 0 0
y: 1 0 1 0
z: 1 0 0 1
```

where x, y, z are data instances, and a, b, c, d are features. In most clustering algorithms, features are assumed to be independent to each other, or their dependencies are ignored. So in the example, x is equally likely clustered with y or z, because the similarity between x and y, and x and z are the same (based on the Euclidean distance, for example). However if we have a priori, general knowledge about the features that b's domain is more similar to that of c than to d, it is better to cluster x and y instead of x and z, because the $\{x, y\}$ cluster is "tighter" than the $\{x, z\}$ cluster with respect to the domains of the features.

Feature domain similarity can be obtained from any linguistic resources (but should be external to the dataset(s) to be used in the experiments). In this work, we used *javasimlib*:[2] a Java-based tool which computes the similarity between words (or synsets) over the WordNet hierarchies based on an information theoretic metric [16]. Given two words or synsets, *javasimlib* returns a value between 0 and 1, where 1 indicates the highest similarity. For each noun, we considered the top (at most) two senses in the calculation in order to avoid spurious results arising from rare or unimportant senses encoded in WordNet. Then we defined the similarity between two nouns as the maximum value between their top two senses:

$$fdsim(a,b) = \arg \max_{i,j \in \{1,2\}} jsim(a\#i, b\#j)$$

where a, b are nouns, $fdsim(a,b)$ denotes the feature domain similarity between a and b, $a\#i$ is the ith sense of a, and $jsim(x,y)$ is the similarity between two synsets x and y returned by *javasimlib*. Then we computed all pairwise similarities between the nouns/features and stored them in a matrix.

Next, we defined a notion of *domain tightness* for a single word vector. This metric indicates how "tight" the vector is with respect to the domains of its features (i.e., the degree of domain *homogeneity*). For a given word vector v, the domain tightness of v, denoted $dt(v)$, is define as:

$$dt(v) = \frac{1}{n} \sum_{a,b \in F; a \neq b} \frac{v(a) + v(b)}{2} fdsim(a,b)$$

where F is the set of features used to represent v, $v(x)$ is the value in v for the feature $x \in F$, $fdsim(a,b)$ is the feature domain similarity between a and b described above, and n is the number of pairwise combinations of the features in F where their feature-values are greater than zero (i.e., n = count($\langle$a,b$\rangle$: $v(a) > 0$ and $v(b) > 0$)). Thus it is the average feature domain similarity between two non-zero features in v, weighted by the average of the two feature-values.

Finally, we defined a new similarity metric between two word vectors that incorporates the feature domain similarity, which we call *dsSim*, as follows. For given word vector $v1$ and $v2$, the similarity between them, denoted $dsSim(v1, v2)$ is:

$$dsSim(v1, v2) = w_0 \times cos(v1, v2) + w_1 \times dt(v3)$$

where $cos(v1, v2)$ is the cosine between $v1$ and $v2$, $v3$ is a vector in which $v1$ and $v2$ are merged (i.e., $v3 = v1 + v2$), and w_0, w_1 are weights which sum up to 1. We used $w_0 = 0.95$ and $w_1 = 0.05$ in our experiments. This new measure is essentially based on cosine, added with the domain tightness of the merged vector. We looked at the merged vector, because a cluster centroid is a vector in which all member word vectors are merged (and averaged), and that's precisely what we wanted to improve upon CBC – the domain tightness of the cluster/committee centroids.

[2] Available from http://wordnet.princeton.edu/links#extensions

4.3. The Sense Discovery Algorithm (DSCBC)

In addition to feature domain similarity, we also incorporated a few more ideas which may help derive monosemous committees. One idea is to limit the number of features to be used in the committee centroids. By taking the top k features, only the most salient features are kept (which hopefully are features in one domain), and other insignificant features are discarded. Another idea is to use only the features that appeared with the majority of the words which make up the committee.

Furthermore, we derived the committees incrementally in two steps: first produce committees from the original dataset, then derive the final set of committees from those produced in the first step. This hierarchical scheme aims to find the second or higher-order features (such as those in a transitive similarity relation).

Finally, we present our algorithm named DSCBC (for *Domain Similarity CBC*) in Figure 2. Overall, it applies the (modified) Phase I and II of CBC twice successively. The modifications to the original CBC are: (1') use the new similarity measure *dsSim* instead of cosine to compute the scores of the derived clusters; and (2') remove from committee centroids the features which co-occurred with less than σ percent of the member words or which are not in the top k features. In our experiments, we used $k = 10$ and 6, and $\sigma = 0.8$ and 0.6 in Step 1 and Step 2 respectively. We reduced k from Step 1 to Step 2 in order to refine the salient features. For σ, we used somewhat higher values, especially in Step 1, because the dataset was relatively dense (since word usage is extremely diverse in web pages). All other parts of the algorithm are unchanged from CBC.

Note that DSCBC may still group "warm" and "cold" in the same cluster (or clusters; a given adjective may appear in more than one cluster, as with CBC). However, such a cluster would have a lower *dsSim* value (hence the cluster score) because the domains of temperature and personality are not similar themselves, thus is less likely to be selected as a committee. If it were, features in less dominant domains (i.e., extended meanings) are discarded because only the top k features are retained in the centroid, thereby resulting in a monosemous committee. In this sense, a DSCBC committee is not equivalent to a synset – rather, it is a list of features (abstract nouns) which explicitly describe the se-

Step 1: Derive committees from data.
Apply Phase I and II of CBC to the dataset using:
- *dsSim* in computing avgsim$(c)^{(\mathbf{1}')}$
- the top k features that are also shared by more than σ percent of the words in the committee to represent its centroid.$^{(\mathbf{2}')}$

Step 2: Derive committees from committees.
Repeat Step 1, but with the committees derived in Step 1.

Step 3: Assign words to committees.
Apply Phase III of the CBC algorithm to the dataset using the final set of committees derived in Step 2.

Figure 2. The DSCBC Algorithm

mantic domain(s) of the cluster.

As we mentioned in the previous section, the main contribution of our algorithm is the incorporation of feature domain similarity in clustering. Our algorithm is novel, and different especially from other clustering algorithms which consider features in some way. For example, Rapp [13] uses a statistical technique called *Independent Component Analysis (ICA)* to discover word senses. ICA is similar to Principal Component Analysis (PCA), and finds components (combinations of features) that are independent to each other and have higher-order similarities. Here, the components are essentially equivalent to CBC/DSCBC committees. However, neither ICA nor PCA can separate features that co-occur frequently – those in the first-order similarity (such as temperature and personality), nor do they limit the features used in the components.

In summary, the key to overcoming the polysemous cluster problem is to use externally obtained information about the similarity/dependency between features and incorporate the information in the clustering process. Filtering insignificant features also helps pick out features in the most dominant semantic domain.

5. Evaluation

To evaluate our algorithm DSCBC, we applied it to the dataset described in section 3, and compared the results with other clustering algorithms, in particular CBC, a graph-based clustering (which uses a min-cut partitioning) and K-means. For the graph and K-means algorithms, we used a tool called CLUTO.[3] Also for those two algorithms, we made some modifications so that they do soft-clustering (which assigns an instance to one or more clusters) in order to make them comparable with CBC and DSCBC. To make modifications, we closely followed the way described in [11]: first apply the algorithm to the dataset and obtain clusters, then apply *MK-means* [20] using the centroids of those clusters as the initial centroids. MK-means is a generalized version of the standard K-means algorithm, and assigns each instance to one or more clusters with which it has the similarity greater than a pre-specified threshold. As with K-means, MK-means performs several iterations until a pre-specified number of iterations is reached. In our experiments, we set the maximum number of iterations to be 5, as with [11].

[3] Available at http://glaros.dtc.umn.edu/gkhome/views/cluto

Table 1. Committee Statistics

	Total # comm.	Ave # features	Ave Domain Tightness	(p-value)	Ave Cosine	(p-value)
DSCBC	38	3.3	0.470	(–)	0.010	(–)
CBC	50	4.9	0.196	(*)	0.012	(0.18)
DSGr	38	6.0	0.224	(*)	0.018	(*)
Gr	50	6.0	0.221	(*)	0.022	(*)
DSKmeans	38	6.0	0.209	(*)	0.012	(0.20)
Kmeans	50	6.0	0.198	(*)	0.011	(0.37)

Furthermore, since DSCBC is an extension of CBC with the domain feature similarity, we also implemented the versions of the graph and K-means algorithms which extend the base algorithms in the same way. So in all, we compared a total of six algorithms: DSCBC, CBC, DSGr, Gr, DSKmeans and Kmeans.

5.1. Derived Committees

First we examined the committees derived by the algorithms. Table 1 shows some statistics. DSCBC produced a total of 38 committees, while CBC produced 50 committees. Other DS algorithms (DSGr and DSKmeans) and non-DS algorithms (Gr and Kmeans) were pre-specified to produce the same number of committees as DSCBC and CBC respectively. The average number of (non-zero) features 3.3 by DSCBC and 4.9 by CBC. For all other algorithms, the average number of features was 6 (= k).

The column "Ave Domain Tightness" indicates the average domain tightness (dt) of the committees derived by each algorithm. Here, a higher value means the feature domain similarity of the committees are overall more similar, therefore the committees are less polysemous. As you can see, DSCBC showed the highest tightness (0.47). The results by all other algorithms, including other DS algorithms, were much lower than DSCBC, and the differences were statistically significant, as evidenced by the p-values[4] (where a symbol (*) indicates < 0.05).

The column "Ave Cosine" indicates the average pairwise cosine between the committees for each algorithm. Here,

[4] The p-values shown in Table 1 are obtained by one-sided t-tests against DSCBC.

DSCBC
{smell, aroma}
{appearance, look}
{quantity, amount, number}
{attitude, countenance, nature}
{day, shade, room, light, face, color}
CBC
{taste, smell, scent, aroma}
{beauty, appearance, look, light, color, thing}
{amount, number, time, information, system}
{tone, attitude, voice, word}
{day, sound, face, light, word, thing}

Figure 3. Example DSCBC and CBC Committees

a lower value means the committees are dissimilar to each other, thus scattered well in the feature space. The result shows that DSCBC had the lowest value, although the differences were not statistically significant in some cases. As for the graph and K-means algorithms, the graph in general seems to produce more monosemous but more correlated clusters than K-means.

Figure 3 shows some example committees derived by DSCBC and CBC. Most of the DSCBC committees seem to pick out a single semantic domain fairly well, whereas CBC committees tend to include extremely abstract nouns such as thing and information, and are more polysemous in general. But some of the DSCBC committees are still polysemous clusters. For example, {attitude, countenance, nature} seems to be a mixture of mental state, appearance and inner quality. Also, {day, shade, room, light, face, color} seems to group different domains – probably caused by the polysemy of color or luminance adjectives such as "bright" and "dark".

5.2. Assigned Word Senses

Next we inspected the senses/committees assigned to words. For each word, the algorithms assign one or more committees. Here, each committee should correspond to a sense of the word. To determine whether or not a committee indeed corresponds to a correct sense of a word, we compared it against the sense encoded in WordNet as the gold standards. Then the evaluation was measured with respect to precision and recall.

5.2.1 Evaluation Methodology

To evaluate the assigned senses, we first looked up each of the 1500 adjectives in the dataset in WordNet and obtained its senses/synsets. Then for each synset, we mapped it to its related noun so that we could utilize the WordNet noun hierarchy to compute the similarity/correspondence.[5] For a given synset, we traversed the *attribute* relation encoded in WordNet, for example, "warm#1" → "temperature#1". If the attribute relation was not available, we traversed the *derivationally related form* relation (for de-nominalized noun), for example, "warm#4" → "warmness#1". Word senses which are not indicated with either relation were ignored in the evaluation. The average number of senses (of the 1500 adjectives) which were associated with either relation in WordNet was 1.63 (while the average of the total number of senses was 2.22).[6] Finally, we determined that an automatically derived committee corresponds to the mapped noun synset if the average similarity between the nouns in the committee and the synset is above a threshold:

$$\frac{1}{|c|} \sum_{f \in F(c)} jsim(f, s) \geq \theta$$

where c is a committee, f is an abstract noun in $F(c)$ (the set of (non-zero) features in c), s is the mapped WordNet noun synset, $jsim(f, s)$ is the similarity between f and s obtained through *javasimlib*, and θ is the threshold. For $jsim(f, s)$, the top (at most) two senses were considered for the feature noun f, and the maximum of the two values returned by *javasimlib* ($jsim(f\#1, s)$ and $jsim(f\#2, s)$) was used.

Figure 4 shows the first three senses assigned/discovered by DSCBC for "cold", "flat" and "democratic". Indicated next to each word string are the number of senses assigned by DSCBC and the number of synsets associated with the two relations in WordNet for the word respectively.

For the evaluation measures, we computed the *precision* of an adjective a as the ratio of the correctly assigned committees of all committees assigned by the algorithm for a. For example, if an algorithm assigns a total 5 senses to a word and 3 of them corresponded to some WordNet sense of the word (including cases when multiple assigned senses corresponded to the same WordNet sense), the precision is 0.6. The precision of an algorithm was the average precision of all adjectives in the dataset.

Then the *recall* of an adjective a was computed as the ratio of the correctly discovered senses of all senses (with either the *attribute* or *derivationally related form* relation) encoded in WordNet for a. So for example, if WordNet encodes either relation to 4 senses for a word and 2 of them were correctly discovered by an algorithm, the recall is 0.5. This recall actually is a tough measure, because the coverage of our dataset is much more limited than WordNet. But

[5] In WordNet, adjectives are not organized hierarchically; instead they are simply categorized into two large groups (descriptive and relational adjectives).

[6] Though this number may sound rather low, in the 1500 adjectives, 723 (= 48%) of them had more than one sense in WordNet, which is quite high.

"cold" [DSCBC: 3, WN: 6]
{complexion, color, face}
{attitude, countenance, nature}
{smell, aroma}
"flat" [DSCBC: 4, WN: 9]
{complexion, color, face}
{figure, form, body, pattern, area, system}
{surface, region}
"democratic" [DSCBC: 8, WN: 3]
{framework, concept, approach, idea, model}
{tendency, view, nature}
{figure, form, body, pattern, area, system}

Figure 4. Example Sense Assignments by DSCBC

Table 2. Precision, Recall and F-measure (when θ = 0.25)

	Precision	Recall	F-measure
DSCBC	0.397	0.397	0.397
CBC	0.304	0.264	0.283
DSGr	0.214	0.251	0.231
Gr	0.192	0.202	0.197
DSKmeans	0.254	0.213	0.232
Kmeans	0.169	0.162	0.166

we thought this measure could provide some indication on the coverage of the algorithms and be utilized to rank the algorithms. The recall of the algorithm was the average recall of all adjectives in the dataset.

5.2.2 Results

Table 2 shows the precision, recall and F-measure for the English dataset when θ = 0.25. The F-measure was computed as standard:

$$F = \frac{2RP}{R+P}$$

where R is the recall, P is the precision. As you can see, DSCBC produced the highest precision as well as recall. Also, the DS-extended algorithms performed considerably better than their non-DS counterparts for both precision and recall – verifying the positive effects made by the incorporation of domain feature similarity. Also notice CBC showed a better performance over other standard algorithms (graph and K-means), including their DS-extended versions. This verifies that CBC is indeed an effective tool for word sense discovery, and that using CBC as the base algorithm for extension in our work was a good choice.

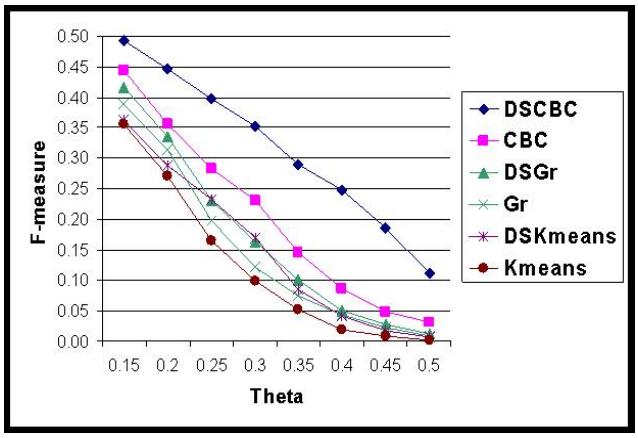

Figure 5. F-measure for varying θ

To investigate further, we also inspected the performance of the algorithms with different values of the threshold θ, since the determination of a correct assignment is dependent on this value. As θ is raised, the precision and recall (thus F-measure) will decrease, because only the higher correspondence values are considered as correct. Figure 5 shows the F-measure for varying θ for all algorithms. DSCBC has the highest F-measure consistently for all θ by a large margin. The DS algorithms are showing a better performance over their non-DS counterparts here as well for all θ, although the differences (between DSGr and Gr, and DSKmeans and Kmeans) are not as dramatic as the difference between DSCBC and CBC.

5.3. Discussions

One thing to note about the sense assignment results is that precision is overall rather low. The reason would probably be because we examined only a subset of the WordNet senses (those which are mapped to nouns). So even when an assigned sense indeed corresponded to a WordNet sense/synset but if the synset is not mapped to a noun in WordNet, it ended up having no matches, thus was considered incorrect in the calculation of precision.

Note that, since the features/nouns are not disambiguated in our dataset, the problem of lexical ambiguity does exist. For example, homonyms (words with multiple meanings in different domains, e.g. "nature" as a quality or the natural world) could in effect "pull" data instances/adjectives in different domains closer, thereby potentially producing polysemous clusters. However in our inspections, lexical ambiguity of features was not so serious – probably because the collocations extracted from the web were already dense. Rather, the most outstanding problem was still due to the polysemy of adjectives, for example the committee men-

tioned earlier, {day, shade, room, light, face, color}.

6. Conclusions and Future Work

In this paper, we presented a new algorithm for discovering word senses for polysemous words, and showed improved results over other clustering algorithms. By incorporating the feature domain similarity in the clustering process, the algorithm produces clusters which are more monosemous and homogeneous with respect to the feature domains.

For future work, an immediate task would be to do a manual evaluation in order to see if the results by the automatic evaluation indeed correlate with human intuitions. Another task would be to investigate exactly which element(s) in the algorithm contributed the most to the improved performance in the current work. Our preliminary inspection indicates the feature domain similarity made more significant contributions than feature filtering.

We would also like to apply the algorithm to other datasets. In fact, we have just completed a preliminary experiment with Japanese adjectives. We applied the same six algorithms to a dataset which contains instances of Japanese adjectives with modifying nouns extracted from newspaper articles [5], and manually compared the derived committees with the senses listed in a conventional dictionary called *Daijirin*. The results so far indicate our algorithm produced better results with respect to precision and recall as well as relative ranking. In addition to adjectives, we would also like to apply the algorithm to other polysemous parts of speech such as verbs. For example, it would be interesting to apply our algorithm to the German verb dataset used in [18] and compare the results.

To extend the current work, we are planning to use the automatically derived committees to investigate the polysemy of adjectives, as we had mentioned earlier in section 2. Since the derived committees are essentially semantic classes of adjectives, by clustering adjectives based on them, we can obtain various patterns of adjectival polysemy defined at an abstract level, such as temperature-personality-color. By inspecting these polysemy patterns, we can study how various adjectival meanings extended – polysemy, metonymy and metaphor of adjectives – attested in real data.

Finally, we would like to investigate the usefulness of the derived word senses in practical tasks and applications such as word sense disambiguation and question-answering.

References

[1] G. Boleda, T. Badia, and E. Batlle. Acquisition of Semantic Classes for Adjectives from Distributional Evidence. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, 2004.

[2] T. Brants and A. Franz. *Web 1T 5-gram Version 1*. Linguistic Data Consortium, 2006. Catalog No. LDC2006T13, ISBN 1-58563-397-6.

[3] V. Hatzivassiloglou and J. Wiebe. Effects of Adjective Orientation and Gradability on Sentence Subjectivity. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, 2000.

[4] D. Hindle. Noun Classification from Predicate-argument Structures. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL-90)*, 1990.

[5] H. Isahara and K. Kanzaki. Lexical Semantics to Disambiguate Polysemous Phenomena of Japanese Adnominal Constituents. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, 1999.

[6] J. Justeson and S. Katz. Principled Disambiguation: Discriminating Adjective Senses with Modified Nouns. *Computational Linguistics*, 21(1):1 – 27, 1995.

[7] M. Lapata. A Corpus-based Account of Regular Polysemy: The Case of Context-sensitive Adjectives. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2001.

[8] J. Levi. *The Syntax and Semantics of Nonpredicating Adjectives in English*. PhD thesis, University of Chicago, 1975.

[9] D. Lin and P. Pantel. Induction of Semantic Classes from Natural Language Text. In *Proceedings of the 7th ACM Conference on Knowledge Discovery and Data Mining (KDD-01)*, 2001.

[10] G. Miller. WordNet: An Online Lexical Database. *International Journal of Lexicography*, 3(4), 1990.

[11] P. Pantel and D. Lin. Discovering Word Senses from Text. In *Proceedings of the 8th ACM Conference on Knowledge Discovery and Data Mining (KDD-02)*, 2002.

[12] J. Pustejovsky. *The Generative Lexicon*. The MIT Press, 1995.

[13] R. Rapp. Mining Text for Word Senses Using Independent Component Analysis. In *Proceedings of the 4th SIAM International Conference on Data Mining*, 2004.

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

[15] H. Schmid. *English Abstract Nouns as Conceptual Shells*. Mouton de Gruyter, Germany, 2000.

[16] N. Seco, T. Veale, and J. Hayes. An Intrinsic Information Content Metric for Semantic Similarity in WordNet. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-04)*, 2004.

[17] T. Takahashi. Various Phases Related to the Part-whole Relationship Investigated in Sentences. *Studies in the Japanese Language*, 103:1–16, 1975. in Japanese.

[18] S. Walde. Experiments on the Automatic Induction of German Semantic Verb Classes. *Computational Linguistics*, 32(2):159 – 194, 2006.

[19] J. Wiebe. Learning Subjective Adjectives from Corpora. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000)*, 2000.

[20] S. Zhong and J. Ghosh. A Unified Framework for Model-based Clustering. *Journal of Machine Learning Research*, 4:1001–1037, 2003.