Supplementary Material:
Clustering Paraphrases by Word Sense

Anne Cocos and Chris Callison-Burch
Computer and Information Science Department, University of Pennsylvania

1 Overview

This document provides additional detail on our similarity metric calculation, clustering algorithm implementation, and CrowdCluster reference cluster development. We also provide full evaluation results across the entire range of our experiments, a selection of sense clusters output by our methods, and example content of our WordNet+ and CrowdCluster paraphrase sets.

2 Computing Similarity Metrics

Unless otherwise noted, we calculate all similarity metrics using PPDB 2.0 data (Pavlick et al., 2015).

2.1 PPDB 2.0 Score

The PPDB 2.0 Score is defined for every pair of words with a paraphrase relationship in PPDB, and in our data set takes values between 1.3 and 5.6. PPDB 2.0 does not provide a score for a word with itself, so we set $PPDB_{2.0}Score(i,i)$ to be the maximum $PPDB_{2.0}Score(i,j)$ such that i and j have the same stem. We assume the PPDB 2.0 Score for non-identical word pairs that are not paraphrases in PPDB is 0.

We define $sim_{PPDB.cos}(i,j)$ as follows:

$$sim_{PPDB.cos}(i,j) = \frac{\sum^{v} S(i,v)S(j,v)}{\sqrt{\sum^{v} S(i,v)}\sqrt{\sum^{v} S(j,v)}}$$

where $v \in V$ are words in the vocabulary, and $S(i,j)$ is shorthand for $PPDB2.0Score(i,j)$.

To calculate $sim_{PPDB.JS}(i,j)$, we must first estimate a probability distribution over paraphrases j for each query word i:

$$P_i(j) = \frac{S(i,j)}{\sum^{v} S(i,v)}$$

We can then calculate the Jensen-Shannon divergence between paraphrases i and j based on their probability distributions P_i and P_j:

$$JSD(P_i \parallel P_j) = \frac{1}{2}KL(P_i \parallel M) + \frac{1}{2}KL(P_j \parallel M)$$

where KL is Kullback-Liebler divergence and $M = \frac{1}{2}(P_i + P_j)$. We set $sim_{PPDB.JS}(i,j) = JSD(P_i \parallel P_j)$. Note that $sim_{PPDB.JS}$ is symmetric.

2.2 Distributional Similarity

We rely on WORD2VEC (Mikolov et al., 2013) word embeddings to calculate our distributional similarity metric:

$$sim_{DISTRIB}(i,j) = \frac{V_i \cdot V_j}{\parallel V_i \parallel \parallel V_j \parallel}$$

where V_i is the vector embedding for word i.

We obtain vector embeddings by downloading 300-dimensional pre-trained vectors from the *word2vec* authors.[1] There are some multi-word phrases and British words in our data set with no exact match in the downloaded vector set. For each British word we use the vector for its American equivalent, and for each unmatched multi-word phrase we take the mean of its individual word vectors as the phrase vector.

[1] These vectors were trained on a Google News data set. Download link: `https://code.google.com/p/word2vec/`

2.3 Translations

We define a pairwise word similarity $sim_{TRANS}(i,j)$ calculated using the foreign translations of i and j from bilingual aligned corpora:

$$sim_{TRANS}(i,j) = \frac{\sum^{f} p(f|i)p(f|j)}{\sqrt{\sum^{f} p(f|i)}\sqrt{\sum^{f} p(f|j)}}$$

where f are foreign words or phrases with which English words i or j are aligned, and $p(f|i)$ gives the conditional probability that i translates to f.

In our work we use Spanish and Chinese foreign translations and probabilities drawn from the corpora used to generate the Multilingual PPDB (Ganitkevitch and Callison-Burch, 2014).

2.4 Entailments

PPDB 2.0 gives a predicted entailment relation between every pair of words in the database. Specifically, it provides a relation-specific entailment probability for each defined relation type (*Equivalent, Forward Entailment, Reverse Entailment, Exclusive*, and *Independent*). In our work we use just the *Independent* entailment probability.

Given a symmetric adjacency matrix W for paraphrase set P, we incorporate entailment information by simply multiplying each adjacency matrix entry w_{ij} by $1 - p_{ind}(i,j)$:

$$w_{ij} = \begin{cases} (1 - p_{ind}(i,j))sim_D(i,j) & (i,j) \in \text{PPDB} \\ 0 & \text{otherwise} \end{cases}$$

3 Clustering Algorithm Implementation

3.1 Overview

We use the general process outlined in Algorithm 1 to cluster paraphrases.

Prior to running clustering, we first consolidate the paraphrase set $PP(q)$ for query term q. If two or more words in $PP(q)$ share a stem, we collapse them into a single paraphrase that takes the properties of the collapsed word with the most PPDB links. If the resulting paraphrase set P has less than three paraphrases, we take a rule-based approach to clustering it: If there is only one paraphrase in P, or if there are two paraphrases in P that are linked in PPDB, we return a single cluster. Otherwise we return two clusters with one word each.

Algorithm 1 Clustering Process

Require: Query word q, similarity method sim_S, distance method sim_D, boolean *entail*, clustering method *method*.

```
Retrieve paraphrase set PP(q) of length n from PPDB.
P ← consolidate_stemmed_wordlist(PP(q))
n' = length(P)
if n' = 1 then
    Set clustering C = {{p_0}}
if n' = 2 then
    if (p_0, p_1) ∈ PPDB then
        Set clustering C = {{p_0, p_1}}
    else
        Set clustering C = {{p_0}, {p_1}}
if n' ≥ 3 then
    W ← get_sim_matrix(P, sim_S)
    S, W ← remove_singletons(W)
    D ← (1−get_sim_matrix(P, sim_D))
    if entail then
        W ← W×get_entail_matrix(P)
    if method =spectral then
        C' ← spectral_cluster(P, W)
    else if method =hgfc
        C' ← hgfc_cluster(P, W)
    C ← optimize_silhouette(C', D)
C ← expand_solution(C, P, S)
```

If the resulting paraphrase set P has length n' of three or more, we cluster it based on the specified method. First we calculate the $n' \times n'$ adjacency matrix W using the procedures outlined in Section 2. If the resulting W has any *singleton* rows, i.e. paraphrases with 0 similarity to all other words in P, we remove the corresponding term(s) from P and add them to their own cluster in the final clustering solution. This is to prevent problems in spectral clustering resulting from such singleton rows.

Next, we calculate the distance matrix D used to optimize the number of clusters using the specified method. If we are using entailments, we execute pointwise multiplication on the adjacency matrix as outlined in Section 2.4. We then execute one of our clustering algorithms described in Sections 3.2 and 3.3. The output of each algorithm is a set of possible clusterings with differing granularity. We choose the optimal clustering based on maximizing the Silhouette Coefficient (Rousseeuw, 1987), with the input distance matrix D.

Finally, before returning the final clustering solution, we expand it to include the singleton clusters we removed earlier and the paraphrases consolidated by stem.

3.2 Hierarchical Graph Factorization Clustering

The Hierarchical Graph Factorization Clustering (HGFC) method was developed by Yu et al. (2006) to probabilistically partition data into hierarchical clusters that gradually merge finer-grained clusters into coarser ones. Sun and Korhonen (2011) applied HGFC to the task of clustering verbs into Levin (1993)-style classes. We adopt Sun and Korhonen's implementation of HGFC for our experiments.

Using HGFC, we represent a paraphrase set $P = \{p_i\}_{i=1}^n$ as an undirected graph $G(P, E)$, where vertices correspond to paraphrases in P and edges $E = \{(p_i, p_j)\}$ connect paraphrase pairs that appear in PPDB. We can represent our chosen similarity measure sim_S between word pairs in P by the nonnegative, symmetric adjacency matrix $W = \{w_{ij}\}$ where the weight of each entry, w_{ij}, conveys the similarity for paraphrase pair $sim_S(p_i, p_j)$. We achieved our best results by normalizing the rows of W such that the L2 norm of each row is equal to 1.

The idea behind HGFC is that we can also estimate w_{ij} using the construction of a bipartite graph $K(P, S)$, where one side contains paraphrase nodes p_i from G and the other consists of nodes from $S = \{s_u\}_{u=1}^k$ corresponding to the latent senses. In this construction, no paraphrase pairs $(p_i, p_j) \in P$ are directly connected, but we can estimate their similarity using hops over senses $s_u \in S$. Specifically, the mapping from W to S is done by the $n \times k$ adjacency matrix B, where B_{iu} gives the weight between paraphrase p_i and sense s_u (Yu et al., 2005):

$$w'_{ij} = \sum_{u=1}^{k} \frac{b_{iu} b_{ju}}{\lambda_u} = \left(B \Lambda^{-1} B^T\right)_{ij} \quad (1)$$

Here, $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_k)$ and $\lambda_u = \sum_{i=1}^n b_{iu}$. If the sum of each row in B is 1, then intuitively b_{iu} corresponds to the likelihood that paraphrase p_i belongs to sense s_u. HGFC uses these likelihoods to produce a soft clustering from the paraphrases in P to the senses in S (Zhou et al., 2004).

HGFC uncovers B and Λ by decoupling them with $H = B\Lambda^{-1}$ and minimizing $\ell(W, H\Lambda T^T)$, s.t. $\sum_{i=1}^n h_{iu} = 1$, given the distance function $\ell(\cdot, \cdot)$ between matrices.

Using the divergence distance $\ell(X, Y) = \sum_{ij}(x_{ij} log \frac{x_{ij}}{y_{ij}} - x_{ij} + y_{ij})$, Yu et al. (2006) showed that the following update equations are non-increasing:

$$\tilde{h}_{iu} \propto h_{iu} \sum_j \frac{w_{ij}}{(H\Lambda H^T)_{ij}} \lambda_u h_{ju}; \sum_i \tilde{h}_{iu} = 1 \quad (2)$$

$$\tilde{\lambda}_u \propto \lambda_u \sum_{ij} \frac{w_{ij}}{(H\Lambda H^T)_{ij}} h_{iu} h_{ju}; \sum_u \tilde{\lambda}_u = \sum_{ij} w_{ij}. \quad (3)$$

Finally, having minimized $\ell(W, H\Lambda T^T)$, we can calculate the affinity between senses:

$$\tilde{W}_{uv} = \sum_{i=1}^{n} \frac{b_{iu} b_{iv}}{d_i} = (B^T D^{-1} B)_{uv} \quad (4)$$

where $D = \mathrm{diag}(d_1, \ldots, d_n)$ and $d_i = \sum_{u=1}^k b_{iu}$.

HGFC works iteratively to create clusters of increasingly coarse granularity. In each round l, the previous round's graph $\tilde{W}_{l-1}$ of size $m_{l-1} \times m_{l-1}$ is clustered into m_1 senses using equations 2 to 4. At each level l, we can recover the cluster assignment probabilities for the original $p_i \in P$ from B_l as follows:

$$prob(s_u^{(l)} | p_i) = (D_1^{-1} B_1 D_2^{-1} B_2 D_3^{-1} B_3 \ldots D_l^{-1} B_l)_{iu} \quad (5)$$

We let the algorithm automatically discover the clustering tree structure by setting m_l equal to the number of non-empty clusters from round $l-1$ minus one.

Algorithm 2 HGFC Algorithm (Yu et al. 2006)

Require: Paraphrase set P of size n, adjacency matrix W of size $n \times n$

$W_0 \leftarrow$`normalize`(W)

Build the graph G_0 from W_0, and $m_0 \leftarrow n$

$l \leftarrow 1$

Initialize cluster count $c \leftarrow n$

while $c > 1$ **do**

  $m_l \leftarrow clustercount - 1$

  Factorize G_{l-1} to obtain bipartite graph K_l with the adjacency matrix B_l of size $m_{l-1} \times m_l$ (eq. 2, 3)

  Build graph G_l with adjacency matrix $\tilde{W}_l = B_l^T D_l^{-1} B_l$, where D_l's diagonal entries are obtained by summation over B_l's columns (eq. 4)

  Compute the cluster assignment probabilities $T_l = D_1^{-1} B_1 D_2^{-1} B_2 \ldots D_l^{-1} B_l$ (eq. 5)

  Set c equal to the number of non-empty clusters in T minus one.

Running the HGFC algorithm returns a set of clusterings of increasingly coarse granularity. For each cluster assignment probability matrix T_l we can recover the soft clustering assignment for each input paraphrase p_i using a threshold parameter τ. We simply take the assignment for each p_i to be the set of senses with probability less than τ away from the maximum probability for that p_i, i.e. $\{s_u | abs(T_{iu}^{(l)} - max_v T_{iv}^{(l)}) \le \tau\}$

When finding the optimal cluster granularity, we find the round l whose clustering assignments maximize the Silhouette Coefficient.

3.3 Spectral Clustering

The second clustering algorithm that we use is Self-Tuning Spectral Clustering (Zelnik-Manor and Perona, 2004)[2]. Whereas HGFC produces a hierarchical clustering, spectral clustering produces a flat clustering with k clusters, with k specified at runtime. The Zelnik-Manor and Perona (2004)'s self-tuning method is based on Ng et al. (2001)'s spectral clustering algorithm.

The algorithm is 'self-tuning' in that it enables clustering of data that is distributed according to different scales. For each data point p_i (i.e. each row in W) input to the algorithm, it constructs a local scaling parameter σ_i:

$$\sigma_i = sim(p_i, p_K) \tag{6}$$

where p_K is the K^{th} nearest neighbor of point p_i. Like Zelnik and Perona, we use $K = 7$ in our experiments.

Using local σ_i, we can then calculate an updated affinity matrix $\hat{A}$ based on similarities given in the input W as follows:

$$\hat{A}_{ij} = \begin{cases} \frac{w_{ij}}{\sigma_i \sigma_j} & i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The complete algorithm we use for spectral clustering is described in Algorithm 3.

To find the optimal number of clusters, we first find m, the number of eigenvectors of L with value equal to 1. We then perform spectral clustering on paraphrase set P with $k \in [max(2, m), min(20, n)]$ and find the k which maximizes the Silhouette Coefficient.

4 Crowd clustering

We want reasonable sets of sense-clustered paraphrases against which to evaluate our automatic clustering method. Although WordNet synsets are a well-vetted standard, they are insufficient for the task by themselves because of their limited coverage. Using WordNet alone would only allow us to evaluate our method as applied to the 38% of paraphrases for our target word list in PPDB that intersect WordNet. So instead we combine crowdsourcing and manual review to construct a reasonable human-generated set of sense-clustered paraphrases.

Some of the paraphrase sets in our PPDB XXL dataset contain more than 200 phrases,

[2] Zelnik and Perona also describe a method for automatically determining the number of clusters in their solution. We do not use this part of their algorithm because optimizing the Silhouette Coefficient gave better results for our data.

Algorithm 3 Spectral Clustering Algorithm (Ng et al. 2001, Zelnik-Manor and Perona 2004)

Require: Paraphrase set P of size n, adjacency matrix W of size $n \times n$, number of clusters k

1: Compute the local scale σ_i for each paraphrase $p_i \in P$ using Eq. 6
2: Form the locally scalled affinity matrix $\hat{A}$, where $\hat{A}_{ij}$ is defined according to Eq. 7
3: Define D to be a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} \hat{A}_{ij}$ and construct the normalized affinity matrix $L = D^{-1/2}\hat{A}D^{-1/2}$.
4: Find $x_1, \ldots, x_k$, the k largest eigenvectors of L, and form the matrix $X = [x_1, \ldots, x_k] \in \mathbb{R}^{n \times k}$.
5: Re-normalize the rows of X to have unit length yielding $Y \in \mathbb{R}^{n \times K}$.
6: Treat each row of Y as a point in $\mathbb{R}^k$ and cluster via k-means.
7: Assign the original point p_i to cluster c if and only if the corresponding row i of the matrix Y was assigned to cluster c.

making it unreasonable to ask a single worker to cluster an entire paraphrase set in one sitting. Instead, we take an iterative approach to crowd clustering by asking individual workers to sort a handful of new paraphrases over multiple iterations. Along the way, as workers agree on the placement of words within sense clusters, we add them to a 'crowd-gold' standard. In each iteration, workers can see the most up-to-date crowd gold clustering solution and are asked to sort new, unclustered paraphrases within it.

4.1 Iterative Clustering Methodology

4.1.1 General overview

Each clustering iteration t includes a *sort* phase in which workers are presented with a list of m unsorted paraphrases $U^t = \{u_1^t, u_2^t ... u_m^t\}$ for a single target word w, and a partial sense clustering solution $C^{t-1} = \{c_1^{t-1}, c_2^{t-1} ... c_k^{t-1}\}$ as generated in previous iterations. The initial round is unseeded, with $C^0 = \emptyset$. Workers are asked to sort all unsorted words u_i^t by adding them to one or more existing clusters $c_{j \leq k}^t$ or new clusters $c_{j>k}^t$. For each target word, n workers sort the same list U^t in each iteration. We add a word u_i^t to the crowd clustering solution C^tif at least $\tau \times n$ workers agree on its placement, where τ is a threshold parameter.

4.1.2 Consolidating Worker Results

When workers add unsorted words to an existing cluster $c_{j \leq k}$, it is easy to assess worker agreement; we can simply count the share of workers who add word u_i to cluster c_j. But when workers add words to a new cluster, we must do additional work to align the j's between workers.

For unsorted words added to new clusters, we consolidate worker placements in iteration t by creating a graph G with a node for each $u_i \in U^t$ added by any worker to a new cluster $c_{j>k}$. We then add weighted edges between each pair of nodes u_i and u_i' in G by counting the number of workers who sorted u_i and u_i' together in some new cluster. Finally we remove edges with weight less than $\tau \times n$ and take the resulting biconnected components as the set of newly added clusters $C^t \setminus C^{t-1}$.

For quality control, we introduce a 'bogus' word that is obviously not a paraphrase of any word in U^t in each round. We ask workers to identify the bogus word and place it in a trash bin. We ignore the results of workers who fail this quality control measure at least 75% of the time.

4.1.3 Merge Phase

We find qualitatively that consolidating clusters based on biconnected components generates overlapping but incomplete clusters after several iterations. So we include a *merge* phase after every third clustering iteration that enables workers to merge clusters from C^{t-1} before sorting new words into C^t. As with the sorting phase, we merge clusters c_{t-1} and c_{t-1}' if at least $\tau \times n$ workers agree that they should be merged.

4.2 Final Cleanup

Using our method, the size of clusters is monotonically increasing each iteration. So before we use the final crowd-clustered data set, we manually review its contents and make corrections where necessary. The full set of reference clusters used in our experiments is given in Section 7.

4.3 User Interface

Our user interface (Figure 1) presents each worker with a 'grab bag' of unclustered words for a given target on the left, and a sorting area on the right. Workers are asked to sort all unclustered words by dragging each one into a bin in the sorting area that contains other words sharing the same sense of the target.

We set the maximum size of the grab bag to be 10 words. This is based on experimentation that showed worker clustering performance declined when the size of the grab bag was larger.

5 Full Results

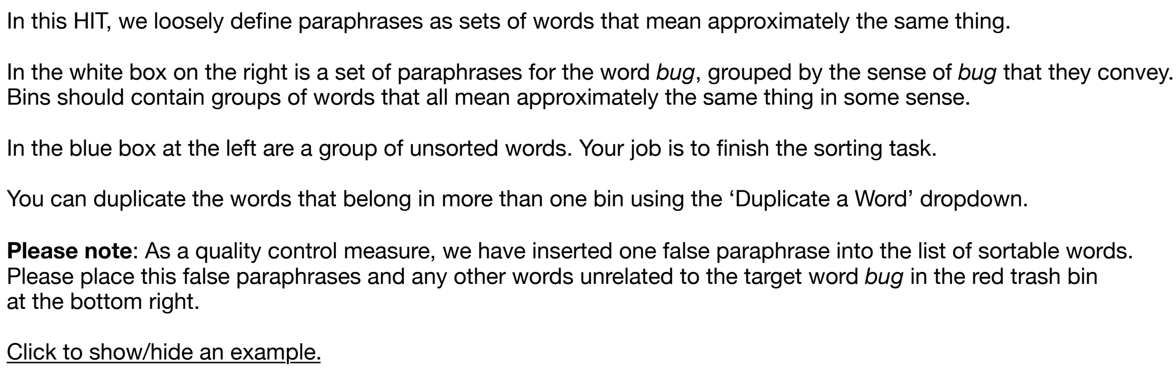
Full results for all experiments are given in Tables 1 and 2. The results given in columns *WordNet+* and *CrowdClusters* indicate the appropriate metric's weighted average across all query words for that set of reference clusters. The result for each query term is weighted by its number of reference classes.

6 Example Clusters

Further examples of the clusters output by our algorithms are given in Figure 2.

7 Reference Sense Clusters

Tables 3 and 4 provide reference clusters for 10 example query words from the WordNet+ and CrowdClusters data sets respectively.



In this HIT, we loosely define paraphrases as sets of words that mean approximately the same thing.

In the white box on the right is a set of paraphrases for the word *bug*, grouped by the sense of *bug* that they convey. Bins should contain groups of words that all mean approximately the same thing in some sense.

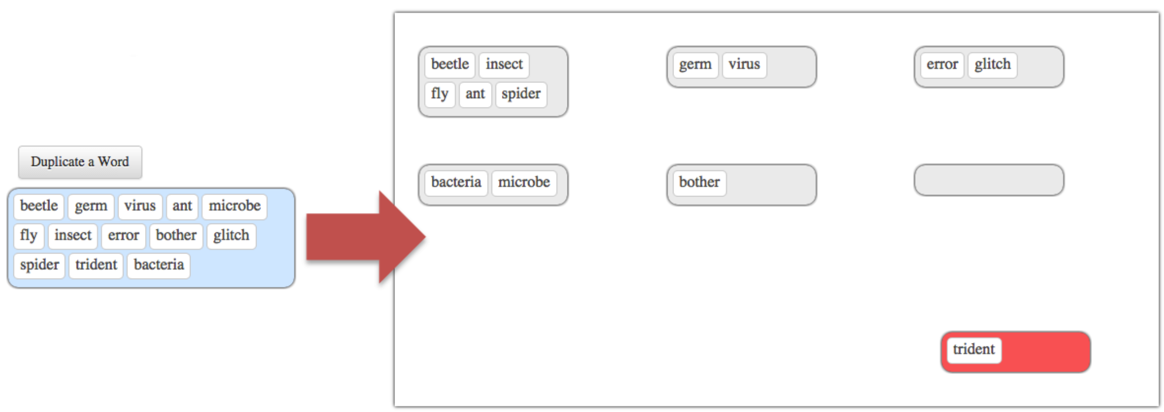
In the blue box at the left are a group of unsorted words. Your job is to finish the sorting task.

You can duplicate the words that belong in more than one bin using the 'Duplicate a Word' dropdown.

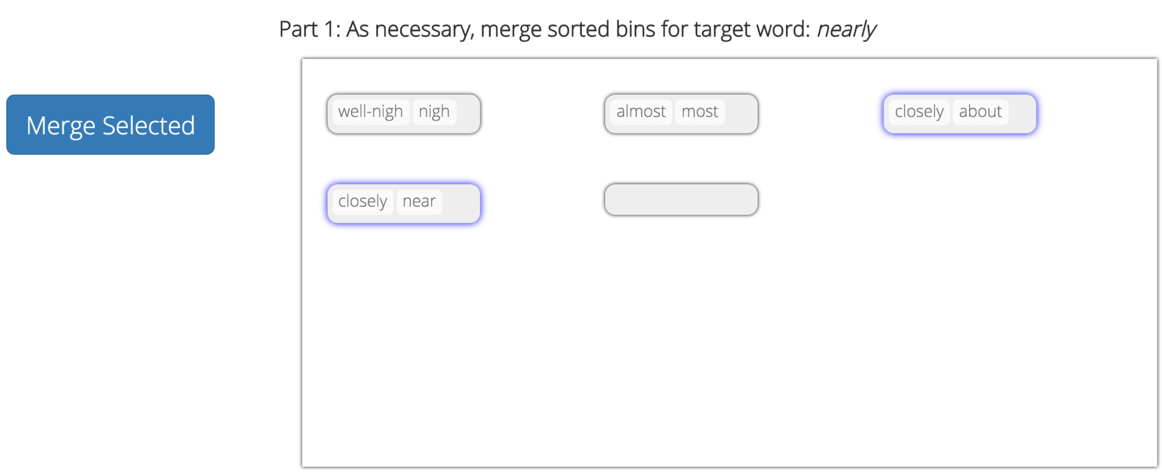
Please note: As a quality control measure, we have inserted one false paraphrase into the list of sortable words. Please place this false paraphrases and any other words unrelated to the target word *bug* in the red trash bin at the bottom right.

Click to show/hide an example.

(a) Sorting user interface instructions to workers.



(b) Sorting user interface.



(c) Merge user interface.

Figure 1: Amazon Mechanical Turk user interface for crowdsourcing reference clusters.

Table 1: HGFC Clustering Results

SimMethod	Choose K Method	Entailments?	Metric	WordNet+	CrowdClusters
PPDB2.0Score	PPDB2.0Score	False	F-Score	0.3497	0.4571
			V-Measure	0.3906	0.4731
		True	F-Score	0.3504	0.4594
			V-Measure	0.3946	0.4681
	$sim_{PPDB.cos}$	False	F-Score	0.3627	0.4979
			V-Measure	0.3947	0.4797
		True	F-Score	0.3539	0.4897
			V-Measure	0.3929	0.4395
	$sim_{PPDB.js}$	False	F-Score	0.3667	0.4737
			V-Measure	0.3899	0.4346
		True	F-Score	0.3550	0.4969
			V-Measure	0.3896	0.4387
	$sim_{DISTRIB}$	False	F-Score	0.3528	0.4893
			V-Measure	0.3332	0.3755
		True	F-Score	0.3587	0.5095
			V-Measure	0.3375	0.3989
	sim_{TRANS}	False	F-Score	0.3494	0.4336
			V-Measure	0.3571	0.3413
		True	F-Score	0.3562	0.4390
			V-Measure	0.3654	0.3502
$sim_{PPDB.cos}$	PPDB2.0Score	False	F-Score	0.3213	0.5007
			V-Measure	0.3256	0.3198
		True	F-Score	0.3465	0.4634
			V-Measure	0.3465	0.4280
	$sim_{PPDB.cos}$	False	F-Score	0.2828	0.4336
			V-Measure	0.4755	0.4569
		True	F-Score	0.3280	0.4425
			V-Measure	0.4548	0.4754
	$sim_{PPDB.js}$	False	F-Score	0.3045	0.4165
			V-Measure	0.4999	0.4622
		True	F-Score	0.3350	0.4691
			V-Measure	0.4187	0.4706
	$sim_{DISTRIB}$	False	F-Score	0.2977	0.4772
			V-Measure	0.3794	0.3270
		True	F-Score	0.3381	0.4422
			V-Measure	0.3662	0.3498
	sim_{TRANS}	False	F-Score	0.3158	0.4102
			V-Measure	0.3373	0.3083
		True	F-Score	0.3276	0.4168
			V-Measure	0.3642	0.3148

Continued…

Table 1: HGFC Clustering Results (continued)

SimMethod	Choose K Method	Entailments?	Metric	WordNet+	CrowdClusters
$sim_{PPDB.JS}$	PPDB2.0Score	False	F-Score	0.3222	0.4754
			V-Measure	0.3045	0.3482
		True	F-Score	0.3530	0.4570
			V-Measure	0.3703	0.4340
	$sim_{PPDB.cos}$	False	F-Score	0.2839	0.4191
			V-Measure	0.4728	0.4799
		True	F-Score	0.3357	0.4365
			V-Measure	0.4457	0.4595
	$sim_{PPDB.js}$	False	F-Score	0.2952	0.3942
			V-Measure	0.4659	0.4703
		True	F-Score	0.3341	0.4452
			V-Measure	0.4391	0.4451
	$sim_{DISTRIB}$	False	F-Score	0.3009	0.4811
			V-Measure	0.3469	0.3535
		True	F-Score	0.3435	0.4781
			V-Measure	0.3563	0.3500
	sim_{TRANS}	False	F-Score	0.3104	0.4026
			V-Measure	0.3114	0.3651
		True	F-Score	0.3247	0.4191
			V-Measure	0.3535	0.3197
$sim_{DISTRIB}$	PPDB2.0Score	False	F-Score	0.2324	0.4476
			V-Measure	0.5261	0.1822
		True	F-Score	0.3311	0.5005
			V-Measure	0.4617	0.4697
	$sim_{PPDB.cos}$	False	F-Score	0.2300	0.4373
			V-Measure	0.5548	0.2467
		True	F-Score	0.3098	0.4920
			V-Measure	0.4724	0.4429
	$sim_{PPDB.js}$	False	F-Score	0.2476	0.4526
			V-Measure	0.4370	0.2681
		True	F-Score	0.3179	0.4847
			V-Measure	0.4935	0.4807
	$sim_{DISTRIB}$	False	F-Score	0.2170	0.3925
			V-Measure	0.5751	0.3977
		True	F-Score	0.2972	0.4663
			V-Measure	0.4905	0.3744
	sim_{TRANS}	False	F-Score	0.2430	0.4036
			V-Measure	0.4942	0.3057
		True	F-Score	0.2957	0.4144
			V-Measure	0.4254	0.4056

Continued...

Table 1: HGFC Clustering Results (continued)

SimMethod	Choose K Method	Entailments?	Metric	WordNet+	CrowdClusters
sim_{TRANS}	PPDB2.0Score	False	F-Score	0.2943	0.4593
			V-Measure	0.2271	0.1530
		True	F-Score	0.3105	0.4587
			V-Measure	0.3094	0.4566
	$sim_{PPDB.cos}$	False	F-Score	0.2969	0.4663
			V-Measure	0.2987	0.2300
		True	F-Score	0.2923	0.4735
			V-Measure	0.3925	0.4353
	$sim_{PPDB.js}$	False	F-Score	0.3027	0.4581
			V-Measure	0.2862	0.1976
		True	F-Score	0.3001	0.4830
			V-Measure	0.3563	0.4340
	$sim_{DISTRIB}$	False	F-Score	0.3001	0.4617
			V-Measure	0.2390	0.2267
		True	F-Score	0.2996	0.4624
			V-Measure	0.3011	0.3367
	sim_{TRANS}	False	F-Score	0.2323	0.3781
			V-Measure	0.4748	0.3106
		True	F-Score	0.2620	0.3887
			V-Measure	0.4095	0.3435

Table 2: Spectral Clustering Results

SimMethod	Choose K Method	Entailments?	Metric	WordNet+	CrowdClusters
PPDB2.0Score	PPDB2.0Score	False	F-Score	0.3268	0.4304
			V-Measure	0.5534	0.5046
		True	F-Score	0.3292	0.4312
			V-Measure	0.5497	0.5326
	$sim_{PPDB.cos}$	False	F-Score	0.3454	0.4865
			V-Measure	0.4698	0.4881
		True	F-Score	0.3517	0.4856
			V-Measure	0.4731	0.4983
	$sim_{PPDB.js}$	False	F-Score	0.3462	0.4858
			V-Measure	0.4556	0.4886
		True	F-Score	0.3510	0.4837
			V-Measure	0.4652	0.4946
	$sim_{DISTRIB}$	False	F-Score	0.3494	0.5067
			V-Measure	0.4452	0.4796
		True	F-Score	0.3570	0.5093
			V-Measure	0.4513	0.4812
	sim_{TRANS}	False	F-Score	0.3231	0.4279
			V-Measure	0.4240	0.4287
		True	F-Score	0.3274	0.4527
			V-Measure	0.4330	0.4330
$sim_{PPDB.cos}$	PPDB2.0Score	False	F-Score	0.3430	0.4888
			V-Measure	0.4823	0.4535
		True	F-Score	0.3317	0.4526
			V-Measure	0.5290	0.4803
	$sim_{PPDB.cos}$	False	F-Score	0.3175	0.4166
			V-Measure	0.5594	0.5244
		True	F-Score	0.3396	0.4635
			V-Measure	0.5019	0.4426
	$sim_{PPDB.js}$	False	F-Score	0.3176	0.4115
			V-Measure	0.5354	0.5053
		True	F-Score	0.3357	0.4660
			V-Measure	0.4793	0.4265
	$sim_{DISTRIB}$	False	F-Score	0.3381	0.4639
			V-Measure	0.4703	0.5018
		True	F-Score	0.3476	0.4811
			V-Measure	0.4224	0.4115
	sim_{TRANS}	False	F-Score	0.3204	0.4940
			V-Measure	0.4069	0.3706
		True	F-Score	0.3234	0.4437
			V-Measure	0.4089	0.3371

Continued...

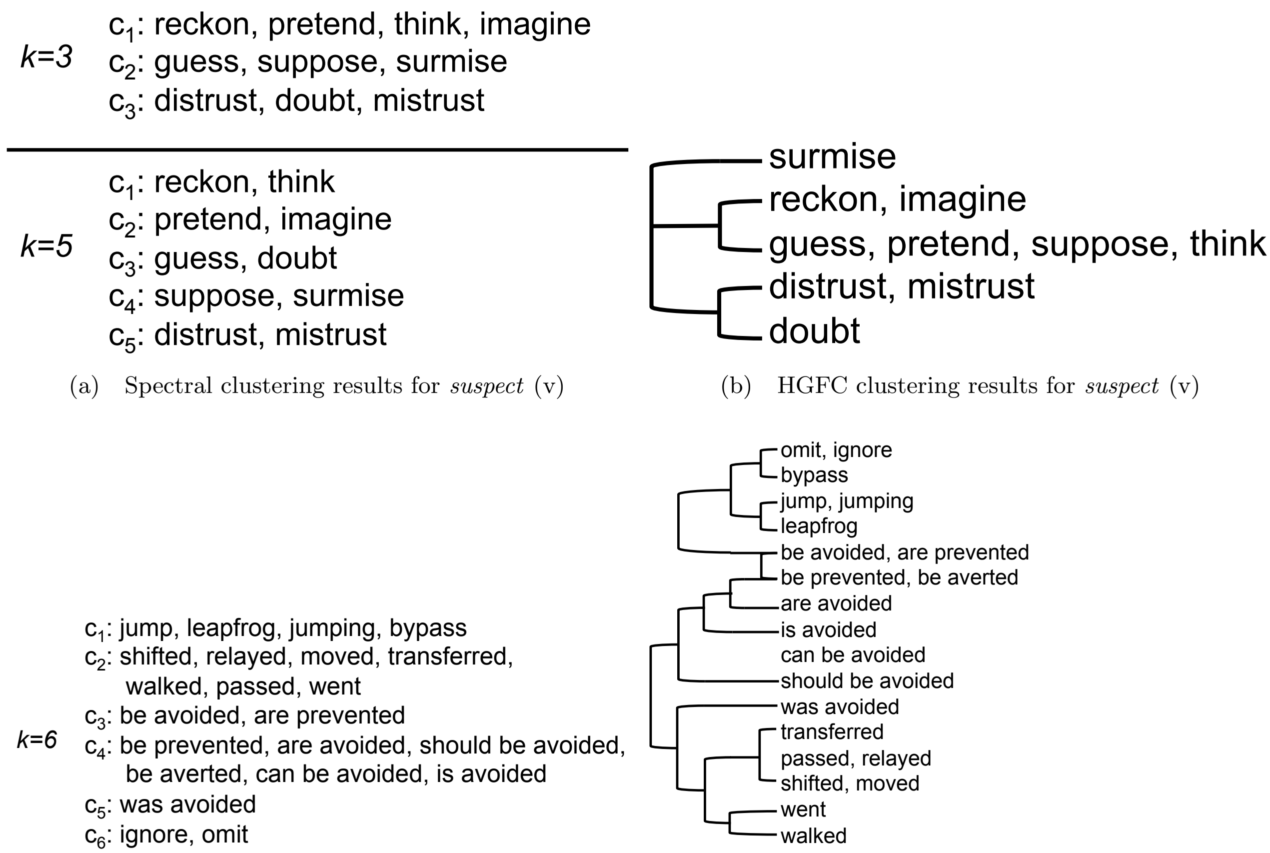
Table 2: Spectral Clustering Results (continued)

SimMethod	Choose K Method	Entailments?	Metric	WordNet+	CrowdClusters
$sim_{PPDB.JS}$	PPDB2.0Score	False	F-Score	0.3389	0.4875
			V-Measure	0.4627	0.4560
		True	F-Score	0.3252	0.4385
			V-Measure	0.5206	0.4753
	$sim_{PPDB.cos}$	False	F-Score	0.3084	0.4109
			V-Measure	0.5442	0.5247
		True	F-Score	0.3327	0.4740
			V-Measure	0.4993	0.4509
	$sim_{PPDB.js}$	False	F-Score	0.3035	0.4003
			V-Measure	0.5233	0.4947
		True	F-Score	0.3327	0.4679
			V-Measure	0.4702	0.4423
	$sim_{DISTRIB}$	False	F-Score	0.3285	0.4701
			V-Measure	0.4581	0.4905
		True	F-Score	0.3412	0.4885
			V-Measure	0.4321	0.4065
	sim_{TRANS}	False	F-Score	0.3095	0.4786
			V-Measure	0.3968	0.3385
		True	F-Score	0.3130	0.4550
			V-Measure	0.3955	0.3418
$sim_{DISTRIB}$	PPDB2.0Score	False	F-Score	0.3182	0.5105
			V-Measure	0.4113	0.4587
		True	F-Score	0.3150	0.4454
			V-Measure	0.5241	0.4815
	$sim_{PPDB.cos}$	False	F-Score	0.3160	0.4436
			V-Measure	0.4805	0.5080
		True	F-Score	0.3436	0.4707
			V-Measure	0.4770	0.4574
	$sim_{PPDB.js}$	False	F-Score	0.3124	0.4658
			V-Measure	0.4547	0.5086
		True	F-Score	0.3472	0.4761
			V-Measure	0.4646	0.4313
	$sim_{DISTRIB}$	False	F-Score	0.2813	0.4244
			V-Measure	0.5137	0.5341
		True	F-Score	0.3367	0.4700
			V-Measure	0.4637	0.4465
	sim_{TRANS}	False	F-Score	0.2984	0.4876
			V-Measure	0.3728	0.3685
		True	F-Score	0.3173	0.4501
			V-Measure	0.3876	0.3531

Continued...

Table 2: Spectral Clustering Results (continued)

SimMethod	Choose K Method	Entailments?	Metric	WordNet+	CrowdClusters
sim_{TRANS}	PPDB2.0Score	False	F-Score	0.2706	0.4461
			V-Measure	0.4154	0.2677
		True	F-Score	0.2617	0.4029
			V-Measure	0.5202	0.4749
	$sim_{PPDB.cos}$	False	F-Score	0.2636	0.4379
			V-Measure	0.4629	0.3650
		True	F-Score	0.2674	0.4231
			V-Measure	0.5107	0.4268
	$sim_{PPDB.js}$	False	F-Score	0.2647	0.4417
			V-Measure	0.4416	0.3655
		True	F-Score	0.2667	0.4242
			V-Measure	0.5106	0.4250
	$sim_{DISTRIB}$	False	F-Score	0.2652	0.4562
			V-Measure	0.4291	0.3655
		True	F-Score	0.2640	0.4476
			V-Measure	0.5158	0.4111
	sim_{TRANS}	False	F-Score	0.2601	0.4441
			V-Measure	0.4180	0.3240
		True	F-Score	0.2584	0.3850
			V-Measure	0.5131	0.4079



(a) Spectral clustering results for *suspect* (v) (b) HGFC clustering results for *suspect* (v)

(c) Spectral clustering results for *skip* (v) (d) HGFC clustering results for *skip* (v)

Figure 2: Results of clusters output by our HGFC and Spectral Clustering methods.

Table 3: WordNet+ Reference Sense Cluster Examples

Query Term	Sense Clusters
film (n)	c_0: wrap, sheet, wrapping
	c_1: flick, picture, telefilm, show, movie, feature, production, documentary
	c_2: episode, sequence, roll, footage, reel, negative, microfilm
	c_3: cinema
touch (v)	c_0: strike, engage, hit, press, feel, handle
	c_1: handle, deal, care
	c_2: strike, affect, move, stir, get
	c_3: be, reach
	c_4: allude, suggest
	c_5: receive, take, have
	c_6: focus on, relate, pertain, regard, concern, involve, apply, affect, hold, refer
	c_7: disturb, modify, violate, change, alter
	c_8: contact, stick, rub, meet, ring, cover
	c_9: impact, hit, influence, bother, modify, alter, treat, strike, affect, stimulate, change

Continued...

Table 3: WordNet+ Reference Sense Cluster Examples (continued)

Query Term	Sense Clusters
soil (n)	c_0: silt, dirt, subsoil, mud, sand, clay, earth, ground
	c_1: territory
	c_2: farmland, land, sod, bottom, turf, ground, tillage
	c_3: filth, dirt
treat (v)	c_0: feed, provide, cater
	c_1: analyze, relieve, analyse, remedy, administer, medicate, nurse, care for, correct, manipulate, operate
	c_2: touch, touch on, run, refine, process, affect, digest
	c_3: react, respond
	c_4: handle, deal, cover, broach, initiate, address, talk about, discuss
	c_5: present, give
	c_6: criminalize, interact, abuse, handle, nurse
severely (r)	c_0: badly, seriously, gravely
	c_1: hard
	c_2: sternly
dark (n)	c_0: nighttime, night
	c_1: shadow, darkness
	c_2: blackness, black, darkness, night
	c_3: darkness
	c_4: darkness
open (a)	c_0: exposed
	c_1: opened
	c_2: receptive
	c_3: candid
	c_4: loose
	c_5: subject, capable
	c_6: clear
	c_7: unresolved, undecided
	c_8: overt
charge (v)	c_0: require, command, burden
	c_1: blame, indict, accuse
	c_2: shoot, rush
	c_3: entrust, check
	c_4: turn on
	c_5: take, direct
	c_6: appoint, authorize, nominate, create, delegate, designate, assign, make
	c_7: load, reload, fill, recharge
	c_8: provide, recharge
	c_9: change
	c_{10}: transfer, send
	c_{11}: set, determine
	c_{12}: pay
	c_{13}: burden, change

Continued...

Table 3: WordNet+ Reference Sense Cluster Examples (continued)

Query Term	Sense Clusters
	c_{14}: blame, ascribe, impute, assign, attribute
	c_{15}: rush
	c_{16}: lodge, accuse, file
	c_{17}: instruct
	c_{18}: claim, tax, complain
	c_{19}: debit
	c_{20}: assess, account, impose, calculate, invoice, bill, levy
board (n)	c_0: plank
	c_1: table
	c_2: card
	c_3: commission, directorate, committee
	c_4: sheet, snowboard, skateboard, surfboard, scoreboard
	c_5: table
	c_6: surface
	c_7: display
	c_8: dashboard, panel
function (n)	c_0: relation
	c_1: ceremony, affair, occasion, party, celebration
	c_2: purpose, use, role, usefulness, utility
	c_3: procedure
	c_4: duty, capacity, office, part, place, portfolio, position, role, hat

Table 4: CrowdCluster Reference Sense Cluster Examples

Query Term	Sense Clusters
post (n)	c_0: positions, job, occupations, position
	c_1: posting, outpost
	c_2: poste, postal
extended (a)	c_0: extension, extend, expanding, expanded, extending, enlarged, stretched, extensive, expand, increased
	c_1: better, enhanced
	c_2: extending, protracted, stretched, prolonged
let (v)	c_0: continued, remained, retained, had
	c_1: derived, prepared
	c_2: 'm leaving, headed, get going, got to go now, going to do, leaving, leave, be used, got to go
	c_3: shown, saw, showed, demonstrated
	c_4: rented, afforded, hired, rent, rented out, owned
	c_5: dropped, declined
	c_6: forgot, forgotten
	c_7: helped, provided, added, included, offered, gave, awarded
clean (v)	c_0: clean-up, cleanliness, clear, 's clean, get cleaned up, cleansing, wiped clean, cleanse, taken up
	c_1: given up, dropped out
	c_2: is true, potable, drinkable, is healthy, is safe
so (r)	c_0: then, now then, well then, so then
	c_1: yes
	c_2: accordingly, so therefore, therefore, thereby, hence, consequently, thus
	c_3: so too, as well, too
	c_4: very
	c_5: even
pull (v)	c_0: been fired, start shooting, shot, keep firing, been shot, get shot
	c_1: get laid, lay
	c_2: conferred, earned
	c_3: 'm coming over, comes up, 's happening, is arriving, coming through, shows up, comes in, 'm coming in, be drawn, is coming, is on his way, 'm coming up, 'm coming, coming in, 're coming, is happening, coming up, 's coming, comes along, 's coming in, 's coming up
	c_4: be accomplished, be undertaken, can be done, should be done, supposed to do, be achieved
	c_5: learnt, derived, be learned, interpreted, been learned, be derived, be learnt, learned
	c_6: is taken, gone, took, drew, can be drawn, drawn, is extracted, are drawn, moving out, was drawn, withdrew, remove, be taken, is removed, withdraw, are taken, got
charge (n)	c_0: accusation, vs, allegation, allegations, indictment, prosecution
	c_1: taxa, charged, fee, charging, surcharge
	c_2: encumbrances, responsibility, burden

Continued...

Table 4: CrowdCluster Reference Sense Cluster Examples (continued)

Query Term	Sense Clusters
	c_3: capita
	c_4: matters
shot (n)	c_0: shoot, gunshot, shootings, shooting
	c_1: shooter
saint (n)	c_0: sainte, st., santa
run (v)	c_0: been organized, enhanced, being managed, organized, acted, structured, designated, owned, conducted, administrated, organised, served, worked
	c_1: am leaving, 're going away, 'm going now, be going, checking out, was going, is going, are going, 'm running
	c_2: are complete, finished, executed, planned, exhausted, bound, can be done
	c_3: be pursued, being pursued
	c_4: commenced, opened, initiated, championed, introduced, circulated, decreed
	c_5: doing, be performed, functioning, worked, operates
	c_6: run off

References

Juri Ganitkevitch and Chris Callison-Burch. 2014. The multilingual paraphrase database. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014), Reykjavik, Iceland*, pages 4276–4283.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*.

Andrew Ng, Michael Jordan, and Y. Weiss. 2001. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*.

Ellie Pavlick, Johan Bos, Malvina Nissim, Charley Beller, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Lin Sun and Anna Korhonen. 2011. Hierarchical verb clustering using graph factorization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1023–1033. Association for Computational Linguistics.

Kai Yu, Shipeng Yu, and Volker Tresp. 2005. Soft clustering on graphs. In *Advances in neural information processing systems*, pages 1553–1560.

Lihi Zelnik-Manor and Pietro Perona. 2004. Self-tuning spectral clustering. In *Advances in neural information processing systems*, pages 1601–1608.

Dengyong Zhou, Thomas Hofmann, and Bernhard Schölkopf. 2004. Semi-supervised learning on directed graphs. In *Advances in neural information processing systems*, pages 1633–1640.