Clustering word senses from semantic mirroring data

Hampus Lilliehöök, Magnus Merkel
Department of Computer and Information Science, Linköping University
hamster.student.liu.se, magnus.merkel@liu.se

ABSTRACT
In this article we describe work on creating word clusters in two steps. First, a graph-based approach to semantic mirroring is used to create primary synonym clusters from a bilingual lexicon. Secondly, the data is represented by vectors in a large vector space and a resource of synonym clusters is then constructed by performing K-means centroid-based clustering on the vectors. We evaluate the results automatically against WordNet and evaluate a sample of word clusters manually. Prospects and applications of the approach are also discussed.

KEYWORDS: word senses, clustering, semantic mirroring.
1 Introduction

The lack of access to lexical resources of high quality can be seen as a bottleneck for the development of language technology applications of many kinds. Many lexical resources, such as bilingual lexicons and thesauri, contain linguistic data that could be even more useful and interesting if the different sources of data could be combined in a controlled manner.

In this work we look at a way of combining the semantic mirroring method with a vector space-based word clustering approach. The idea is that while semantic mirroring is an excellent starting point for building lexical resources, given a bilingual lexicon or word-aligned parallel corpus, semantic mirroring data could be refined using more standard clustering techniques.

Semantic mirroring means using bilingual resources, such as lexicons or parallel corpora, for identifying semantic relations (e.g. synonymy and hypernymy) between words in a common language. Several experiments have been done, yielding clusters of words related in this manner. Those of importance to this article are Dyvik (2004) and Eldén et al. (2013).

Eldén et al. (2013) developed an application in MATLAB for carrying out a graph-based semantic mirroring procedure.¹ A simplified description of how it is done is as follows: A seed word from language a is looked up in a dictionary, or a translation matrix, yielding a set of words in language b. Each of these are then translated back, generating a set of words in a, including the seed word. Each of these, except the seed word, are translated back and forth once again in the same manner, ignoring words that were not in the first translation set – with the number of translation paths² being counted – yielding a weighted adjacency matrix representing the number of paths between words in the translation matrix. Using measures of connectedness derived from the data generated in the prior step – and spectral graph theory – the adjacency matrix is decomposed into smaller word clusters. By assumption, these clusters represent different word senses – often, but not always related to that of the seed word.

The method has shown promising results, but the effects of combining the results from several seed words have not been examined in detail. There are several reasons why one wants to do this, perhaps most clearly illustrated by the technical limitations of applying the procedure only once. Firstly, the seed word is lost in some clusters from the GBSM application where it should be kept; by unifying similar clusters we aim to retain the seed word. Secondly, it is only when the procedure is repeated that a word can be associated with several senses.³

The aim of this work is to extend the GBSM application used by Eldén et al. (2013) to generate the word clusters for all available seed words, and to unify the results using a vector space model. In effect this means combining the word clusters generated from all seed words into a unified word sense resource, the result of which is evaluated in two steps:


¹The work described in Eldén et al. (2013) is an extension of the implementation in Fagerlund et al. (2010). From here on the graph-based semantic mirroring application is referred to as the GBSM application.
²A translation path links together two words in some language via a shared translation word in another language.
³Which is desired every time two or more words are homonyms.
2. Manually: by examining a randomly selected subset of the output data.

2 Semantic mirroring and graph-based semantic mirroring (GBSM) application

The extraction of semantic data from bilingual resources is motivated linguistically by assumptions regarding the nature of translation. The term “mirroring” refers to the view of a translation as a semantic mirror of the source (Dyvik, 2004). The translation and source could be for example a bilingual dictionary, or an aligned parallel corpus. Semantically related words tend to have overlapping sets of translations in other languages. Words in one language are nodes joining together words in another language through translation. Given that the assumptions are valid, and that the resource is of adequate quality, translation paths between any two nodes (words) in a language indicate that there is some semantic relation between them; they may be synonyms, have similar meanings, or one word may be a subordinate to the other (hyponym). They may also share accidental translation paths, e.g. due to word strings having multiple meanings (homonyms and/or homographs).

Below is an example to give some idea of how the method can be applied. This example uses nouns, but is in other aspects similar to how it has been used in the GBSM application:

The Swedish seed word rätt (noun) is looked up in a dictionary, and the following translations are found:

\{course, dish, meal, justice, law, court\}.

Among the words in this set, we want to find translation paths via words in the Swedish language, so they are all translated back and forth. course has the Swedish translation set

\{lopp, kurs, lärokurs, flöde, fat, gång, stråt, väg\}

of which kurs and fat have translations in the initial translation set; kurs translates to

\{tack, course, class\},

and fat translates to

\{bowl, saucer, plate, barrel, course, dish, platter\}.

Thus, it turns out that course has a translation path to dish, and by assumption they have similar meanings, hence they qualify as a cluster.\(^4\) Further on, it turns out that there is a whole structure of interconnectedness in the initial translation set. The task of the GBSM application is to identify this structure and mathematically determine where its links are weakest, and to decompose the interconnected structure into smaller clusters – each representing a semantic meaning. The translation matrix used by the GBSM application was generated from a subset of Norstedts Swedish – English lexicon, containing only adjectives (Norstedts, 2000).

\(^4\)It also turns out that dish has a translation path to course. This is common, but not necessarily the case at all times.
3 Clustering

By clustering we mean the task of grouping sets of objects into new groups (or clusters) so that the objects in the same cluster are more similar to each other than to those in other clusters.

How clustering is done thus depends largely on the type of data, and for what end the data is being clustered. In most cases the data can be represented as points in some metric space, with data points (or items) having a similarity (or dissimilarity) measurable as a distance between each other in this space (Witten et al., 2011).

Two large categories of clustering are hierarchical clustering and centroid-based clustering. Hierarchical clustering can be either divisive (roughly: top-down) or agglomerative (roughly: bottom-up): Divisive clustering means that the algorithm starts with all data items belonging to a single cluster, with the goal of decomposing it into several clusters. Agglomerative clustering, starts with each data item as its own cluster, successively merging them together. A linkage criterion is used for determining which data items are involved in agglomeration or division (Witten et al., 2011). Three common criteria are single-linkage, complete-linkage and average-linkage. Single-linkage may cause clustering operations to be done when any pair of data items have a high similarity metric. With complete-linkage, all items in a cluster are taken into account; for example maximizing the sum of similarities for each data item in a cluster, against some candidate. With average-linkage, similarity is measured against some mean value of a cluster. When hierarchical clustering is done, a dendrogram is often generated, representing the points in time at which clusters were merged (agglomerated) or decomposed (divided). Selecting a point yields the cluster space at that time.

Centroid-based clustering starts with an assumption about how many clusters the data is to be grouped into (this value is often called $K$). An initial clustering is done, either randomly or according to some heuristic. Next, the centroid items of each cluster are calculated. This can be done arbitrarily but a mean value is commonly used. The centroid items may or may not be actual members of the cluster. Following this, all members of all clusters are associated with their nearest, or most similar centroid, according to a selected similarity metric (for example Euclidean distance). Each data item is then moved to the cluster containing that centroid, and the process is repeated – including recalculation of the centroids – until no more items need to be moved. The cluster space present at this stage is the result of the algorithm.

Clustering algorithms in general suffer from some drawbacks. First of all, they are computationally hard – especially noticeable when dealing with large and/or high-dimensional datasets. Secondly, the numerous configurations that are possible makes them unlikely to produce satisfying results without some supervision, testing and tweaking.

There are also drawbacks related specifically to the different categories of clustering algorithms: For hierarchical clustering, it is hard to algorithmically decide at what point in time to stop the agglomeration or division. For centroid-based clustering, it is hard to know in advance what a good value for $K$ is. Also $K$ needs to be kept relatively small due to time complexity. Further on, the outcome of centroid-based clustering is very sensitive to the state of the initial clustering – the solution is guaranteed to be optimal with regard to this state – but there may exist better, but unreachable global solutions.

\(^5\)One configuration of this algorithm is commonly known as K-means or Lloyd’s algorithm.
4 Extending the GBSM application

The first practical step in this work has been to compile all the required input data. As mentioned earlier, the graph-based semantic mirroring application (GBSM) developed by Eldén et al. (2013) for their experiment has been used for this.

The program has been modified to generate all data at once, in an unsupervised manner; the program body has been included in a loop statement, which terminates after all words in the dictionary have been processed. The interactive features of the application have been replaced by statically or dynamically declared values where appropriate.

One main feature of the program is a loop that divides larger groups (at this stage they are represented as graphs) into smaller ones, until a condition is met. A threshold fiedler value of 0.2 has been used for this. The fiedler value is a measurement for the connectedness of a graph; if the fiedler value is below the threshold the graph is too loosely connected and a cut must be made to separate the graph into two. Calculating the cut that yields two as connected subgraphs as possible is computationally difficult\(^6\) and the fiedler value is a heuristic for this problem.\(^7\)

In effect, sprawly, or loosely connected groups are decomposed into smaller, more tightly connected groups. Groups having a fiedler value equal to, or above the threshold level, are considered good enough, and the decomposition loop terminates.

Output data are all clusters of English adjectives, derived from graphs with three or more nodes\(^8\) associated with the seed words that were used to retrieve them.

Python was used to implement the extended GBSM application, for its ease of use with text data and availability of relevant packages: NumPy, part of the SciPy library (Jones et al., 2001), iPython (Pérez and Granger, 2007), and WordNet (Miller, 1995), used here as part of the NLTK Python library (Bird et al., 2009).\(^9\)

4.1 Dataset

As mentioned earlier, the dataset, or input data, are clusters of English adjectives, some of which can be seen in figure 1. Each section separated by ## represents the result of one iteration in the main loop of the modified GBSM application. The first two items of numerical data are only labels for the clusters and of no significance. What follows afterwards is the seed word followed by lines of data, or clusters, derived from the seed word, each including its fiedler value and word strings.

One important clarification on terminology: The input data consists of word clusters that are to be clustered together by comparing their vectors. The term cluster in this text can thus mean either a word cluster (a set of items, or a vector representing it), or a cluster of clusters (a set of sets, or a set of vectors). In the context of input data analysis or reduction of the dataset, clusters refer solely to word clusters that are present in the input data. In the context of K-means clustering, on the other hand, the data items to be clustered are in fact

---

\(^6\)This is an instance of the minimization problem (Eldén et al., 2013).

\(^7\)For details about how the fiedler value is calculated, see Eldén et al. (2013)

\(^8\)Nodes are words in a common language, that share some translation path(s).

\(^9\)WordNet is often used as a gold standard for evaluation of automated extraction of word sense data ((Cicurel et al., 2006), (Bansal et al., 2012))
 interchangeablerenewablecommutablepermutable

arcticnorthernnorth
Hyperboreanborealnorthwardnorth-polar

cannonproofshotproofbulletproof

**Figure 1:** An excerpt of the dataset generated by the GBSM application, used as input for clustering.

\[
\{c_a = \{w_{a_1}, \ldots, w_{a_z}\}, \\
c_b = \{w_{b_1}, \ldots, w_{b_z}\}, \\
\ldots \\
\{w_1, w_2, \ldots, w_n\} \rightarrow c_j = \{w_{j_1}, \ldots, w_{j_z}\}, \\
c_k = \{w_{k_1}, \ldots, w_{k_z}\}, \\
\ldots \\
c_m = \{w_{m_1}, \ldots, w_{m_z}\}\}
\]

**Figure 2:** A set of possibly overlapping word clusters is yielded from a set of seed words.

clusters themselves, so clustering of data items means the grouping of word clusters into sets.

The final product of this application are *word clusters* that are yielded from merging similar *word cluster clusters*. This is an important property of this project; while the GBSM application groups words into sets, the Python application groups sets of words into sets of sets – which are then merged (see figures 2 and 3).

### 4.1.1 Reductions in the dataset

Reductions in the dataset have been made, in order to both decrease its size and increase its quality. The upper threshold, discarding input data clusters of larger size, has been set to 15.

It is reasonable to argue that word clusters should be no larger; first of all, they are to be merged later on, and are thus to grow even larger. Also, it is assumed that most of the larger ones are large because of unfortunate properties of the GBSM application, rather than valid

\[
\{c_a, c_b, \ldots, c_m\} \rightarrow \{\{c_a, \ldots, c_i\}, \{c_j, \ldots, c_k\}, \ldots, \{c_l, \ldots, c_m\}\}
\]

\[
\{\{w_{a_1}, \ldots, w_{a_z}\}, \ldots, \{w_{i_1}, \ldots, w_{i_z}\}\}, \\
\rightarrow \{w_{j_1}, \ldots, w_{j_z}, \ldots, w_{k_1}, \ldots, w_{k_z}\}, \\
\ldots \\
\{w_{l_1}, \ldots, w_{l_z}, \ldots, w_{m_1}, \ldots, w_{m_z}\}\}
\]

**Figure 3:** Word clusters from the GBSM application are clustered into sets of word clusters. The unique word set of the clustered clusters are considered the result.
semantic properties. Clusters of sizes above 200 are not uncommon. By comparison, the largest synset of adjectives in WordNet is of size 23, and the average size is \( \sim 1.65 \).

We have also chosen to filter duplicate clusters. Duplicate clusters in this sense are word clusters containing the same set of words – i.e. regardless of the seed word, fiedler value or order (they are automatically sorted upon creation). The set of words is the only attribute of a cluster that plays any role in the clustering algorithm (see section 4.3). It should be noted, however, that if duplicate word clusters are present in the clustering algorithm, they do affect it, but they will not add any qualitative features – only quantity to features already present.

### 4.2 Vector space model

Each unique word in the input data is assigned an index in the range \([0, n - 1]\) of natural numbers, \(n\) being the number of unique words in the input data. Each word cluster is assigned in the same manner for the range \([0, m - 1]\), \(m\) being the number of word clusters in the input data. For each of the clusters, a vector \(v\) of length \(n\) is created. \(v_{i,j} = 1\) if word with index \(j\) is a member of the cluster with index \(i\), \(v_{i,j} = 0\) otherwise.

\[
V_{m,n} = \begin{bmatrix}
    [v_{0,0} & v_{0,1} & \cdots & v_{0,n-1}] \\
    [v_{1,0} & v_{1,1} & \cdots & v_{1,n-1}] \\
    \vdots \\
    [v_{m-1,0} & v_{m-1,1} & \cdots & v_{m-1,n-1}]
\end{bmatrix}
\]

Using the reduced input data: \(n = 8832, m = 13691\). The graphic style of the matrix bears some resemblance to the data structures used for its representation. While the matrix is an array of dimensions \(m \times n\), each row vector is in its own an array of dimension \(1 \times n\), and as an element in the matrix array they can be accessed in constant time. For all partitioning and clustering purposes, lists of indices referring to these vectors are the data items, and various lookup functions are used for comparison.

Clustering in general is computationally hard in large and/or high-dimensional datasets. Depending on the configuration (see section 4.1) – some parameters determine which input data to be accepted or discarded – the number of data items, or word clusters in this application is in the range of 13691 – 21093. With an average cluster size in the range of 4.70 – 6.75, we could expect \(K\) to be in the order of around 2200 – 4500. Further on, the dimensionality (in this case equal to the number of unique words, which is the capacity each vector needs to hold) is in the order of around 8800 – 10000. Using K-means directly on this set quickly proved to be intractable.

Some observations were made, however. We found that some sets of word clusters would always be disconnected from the rest. Formally, this means that some sets of vectors can be found, with each vector being orthogonal to all other vectors in the vector space except at least one within its own set.

To give an example, consider the following matrix \(M\) as our vector space:
Analogous to the vector space (section 4.2), indices $a$, $b$, $c$, $d$ and $e$ represent words that may or may not be members of clusters $u$, $v$, $w$, $x$ and $y$. The value $1$ at $M_{u,a}$ means that word $a$ is a member of vector $u$. In this case, vectors $u$ and $v$ are both orthogonal to vectors $w$, $x$ and $y$, but not to each other. Vectors $w$ and $y$ are orthogonal to each other, but neither of them is orthogonal to vector $x$.

Any way of disjoining $\{w, x, y\}$ into two or more sets would entail non-orthogonality between some pair of these sets, hence there is no way of separating them further.

As a result, $M$ can be reduced into the two matrices $M_1'$ and $M_2'$:

$$M_1' = \begin{pmatrix} a & b & c & d & e \\ u & 1 & 0 & 0 & 0 \\ v & 1 & 1 & 0 & 0 \\ w & 0 & 0 & 1 & 1 \\ x & 0 & 0 & 0 & 1 \\ y & 0 & 0 & 0 & 0 \end{pmatrix}, \quad M_2' = \begin{pmatrix} a & b & c & d & e \\ w & 0 & 0 & 1 & 1 \\ x & 0 & 0 & 0 & 1 \\ y & 0 & 0 & 0 & 0 \end{pmatrix}$$

Or even simpler:

$$M_1'' = \begin{pmatrix} a & b \\ u & 1 \end{pmatrix}, \quad M_2'' = \begin{pmatrix} c & d & e \\ w & 1 & 1 \\ x & 0 & 1 \\ y & 0 & 0 \end{pmatrix}$$

The hope was to reduce the vector space into a few roughly equally large subspaces by this principle, each of which would then be input to its own instance of the clustering algorithm, thus reducing the factors of time complexity. Unfortunately, this was only a moderate success. The partitioning of the vector space by this criterion instead yielded one very large subspace of size 13085 and dimensionality 8127 – out of the complete vector space of size 13691 and dimensionality 8832 – plus many smaller ones.

On the positive side, it had a very useful side effect with regard to qualitative, or semantic aspects. Many of the disjoint spaces would in fact be found by the clustering algorithm, although it is not guaranteed. Additionally, since cosine similarity is used as the metric for cluster- or vector comparison, the partitioning of the vector space as described above yields the most dissimilar sets of clusters there are. This operation thus proves to be not only reducing the computational time of the clustering algorithm, but also guarantees improving the result.

The large connected subspace has the size of 13085 out of 15328 vectors (85.34%) when duplicates are allowed, and 13085 out of 13691 vectors (95.57%) when duplicates are forbidden (as in the case described above). Hence, all duplicate clusters appear to occur outside of this connected subspace.

---

10 Explained in section 4.3.
4.3 The K-means algorithm

What remains before evaluation is to perform clustering on the data that can be reduced no further using disjunctive partitioning. The K-means algorithm has been chosen for this.

With hierarchical clustering there is an uncertainty of when to stop the division or agglomeration, while with centroid-based clustering (K-means being of this family), the value $K$, or the expected number of clusters, is pre-determined, and results from different configurations may need to be compared.

Initially, the value $K$ is set, and two arrays of size $K$ are created:

1. Array $C$ for the centroid vectors, which is initially empty
2. Array $D$ containing the partitions of vectors obtained from making $K - 1$ cuts in the data. This is the initial clustering.

Next, the centroid vectors for each of the initial clusters in $D$ are calculated. For each data item, or vector, in each cluster of $D$, its nearest centroid vector in $C$ is calculated by measuring each centroid against the item using cosine, and the item is moved to the cluster having that centroid vector. If no nearer centroid vector is found, then the item is not moved. Next, the centroids are re-calculated and the procedure is repeated until no nearer centroid vector is available for any item. The state of $D$ at this point is the result of the algorithm.

4.3.1 Weaknesses

The complexity of the K-means algorithm increases as a function of the following three factors: 1) the size $N$ of $D$; 2) the value $K$, and 3) the number of iterations $I$ needed to reach a solution.

The computational time complexity of the algorithm is in $\mathcal{O}(N \times K \times I)$. Further on, the computation time for each item-centroid comparison depends on the dimensionality of the vectors, what metric is used and how it is implemented.

4.3.2 Implementation in a word sense context

The initial idea was to use K-means directly on the vector space. The reductions in the input data were done partially for reducing complexity-causing factors, and partially for qualitative reasons. The partitioning of the cluster space did provide further reductions in data size, although not as much as was needed. K-means thus served two purposes:

1. To reduce the large non-disconnected subspace into more manageable subspaces.
2. To perform clustering on each subspace.

In practice this meant that the large interconnected subspace of size 13085 was used as input to the K-means algorithm, with $K = 15$, generating 15 clusters of sizes in the order of around 600 – 1100. Next, K-means was applied on each subspace, with $K$ derived from the number of unique words in that subspace. Figure 4 gives an illustration of this.
4.4 WordNet as gold standard

Using WordNet as a gold standard is not ideal, but it is an extensive resource, performing well at approximating results, and it provides a foothold where the set of options otherwise would be small.

Cicurel et al. (2006) and Bansal et al. (2012) both evaluate word sense clustering results against WordNet. How this is done – i.e. how corresponding WordNet synsets are chosen, what metric is used, and whether WordNet’s hierarchical relations are taken into account – is subject to variety. We have chosen to evaluate our data against WordNet as well, using a method similar to one presented by Cicurel et al. (2006) in their article; one-to-one association. As the name suggests, one WordNet synset is associated and compared to one word cluster.

Our evaluation procedure is as follows: Fifty clusters were selected at random on five intervals of cluster sizes, aiming at an even distribution.

For each of the clusters that contain at least one word that is present also in WordNet, we fetch every WordNet synset of adjectives having at least one word in common with that cluster. The similarities between the word cluster and the synsets are then determined using cosine. We do this in three ways:

1. With words not in WordNet removed from the cluster, plus words not in our resource’s word space removed from each WordNet comparison synset
2. With words not in WordNet removed from the cluster
3. Without removing anything

By doing (1) and (2), we simulate a so-called projection of our clustering onto WordNet’s set of words. In effect, the clusters and synsets that are compared in (1) will always be subsets of the same word space, while in (2) and (3), this property is gradually relaxed, allowing for more dissimilarities between the word spaces of the cluster and the synset.

---

13 This is partly inspired by (Bansal et al., 2012), who project WordNet’s synsets onto their word resource to create reference clusters to evaluate against. Intuitively, one can think of the projection as the answer to “How would WordNet cluster this set of words?”
What one can expect from this is that the similarity score will be best in (1), followed by a gradual decline in (2) and (3).

Regardless of method, the word cluster and the synset are both represented as vectors with dimension equal to the number of words in their union. Differences in case are ignored, and the frequent separation characters “-” and “_” are replaced by spaces, as to normalize inconsistencies between the two resources. For each of the three methods, the synset having the highest similarity with the word cluster is chosen as the associated synset for that cluster.\[^{14}\]

\[
\text{similarity}\{\text{horse, apple, banana}\}, \{\text{apple, banana, aircraft}\}\) = \cos(\theta) = \frac{(1, 1, 1, 0) \cdot (0, 1, 1, 1)}{\| (1, 1, 1, 0) \| \| (0, 1, 1, 1) \|} = \frac{2}{3}
\]

**Figure 5:** The two sets are compared using cosine, in a vector space having the dimension of their union.

Finally, the average similarity scores for the associated synsets are calculated. For all three sets of similarity scores described above, average similarity is calculated not only for the sets respectively, but also for the intervals of cluster sizes. Large clusters are suspected to score less, and this helps illustrate the suggested correlation. The average similarity scores for the entire output dataset are also calculated and presented.

### 4.5 Manual evaluation

As a second step in the evaluation, the authors evaluated the set of fifty word clusters manually. The clusters are the same as in the previous step. In this step, "linguistic intuition" was complemented by the use of dictionaries for verification.\[^{15}\] The clusters were investigated independently by each evaluator, the predominant senses were identified, and then a precision score was calculated, based on the proportion of suggested words sharing the same meaning in relation to the number of words in the cluster. The interpretations and scores were then compared and discussed, and a consensus conclusion was reached.

We are however aware of our limitations as non-native English speakers, and would have preferred assistance of independent and experienced lexicographers, for a more reliable account. However, we believe the results fairly reflect the overall performance. A recurring difficulty in this area is to determine what synonyms really are. Jurafsky and Martin (2009) provide a method of testing for multiple word senses: “We might consider two senses discrete if they have independent truth conditions, different syntactic behavior, and independent sense relations, or if they exhibit antagonistic meanings.” By contrast, this means that synonyms are lexemes that denote the same word sense, and by this reasoning, substituting a word for a synonym should preserve its propositional truth value and syntactic function in a sentence. WordNet (Miller, 1995) states: “Synonyms – words that denote the same concept and are interchangeable in many contexts.” Any two synonyms will, however, carry different connotations. These usually affect the interpretation of the words, and thus in what contexts one might expect to find them.

\[^{14}\]A similarity \( \in (0, 1] \), \( \in \mathbb{R} \) is obtained, where 1 means that the compared sets of words are identical.

5 Results

Out of the 13691 clusters from the GBSM application, 2727 word clusters, or senses, were distinguished by K-means. Figure 6 shows how these are distributed by the criterion number of unique words. 1824 words, out of our total of 8832 unique words, are not in WordNet.

<table>
<thead>
<tr>
<th>Cluster size</th>
<th>Clusters in range</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-4</td>
<td>467</td>
</tr>
<tr>
<td>5-7</td>
<td>629</td>
</tr>
<tr>
<td>8-11</td>
<td>681</td>
</tr>
<tr>
<td>12-17</td>
<td>592</td>
</tr>
<tr>
<td>18-70</td>
<td>358</td>
</tr>
</tbody>
</table>

Figure 6: Distribution of clusters by their number of unique words.

5.1 WordNet as gold standard

Below the results from the first evaluation procedure are shown. Average similarity are the three different measurements described in section 4.4. Since we suspected that clusters of different sizes would not score equally well, scores for the size ranges described earlier have been measured, in addition to the overall score.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2-4</td>
<td>0.765</td>
<td>0.703</td>
<td>0.576</td>
</tr>
<tr>
<td>5-7</td>
<td>0.505</td>
<td>0.505</td>
<td>0.471</td>
</tr>
<tr>
<td>8-11</td>
<td>0.485</td>
<td>0.464</td>
<td>0.438</td>
</tr>
<tr>
<td>12-17</td>
<td>0.468</td>
<td>0.434</td>
<td>0.392</td>
</tr>
<tr>
<td>18-70</td>
<td>0.374</td>
<td>0.352</td>
<td>0.329</td>
</tr>
</tbody>
</table>

Figure 7: Average similarity with nearest WordNet synsets.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>all (50 clusters)</td>
<td>0.515</td>
<td>0.487</td>
<td>0.438</td>
</tr>
<tr>
<td>all (2727 clusters)</td>
<td>0.508</td>
<td>0.481</td>
<td>0.432</td>
</tr>
</tbody>
</table>

Figure 8: Average similarity with nearest WordNet synsets.

As figure 7 suggests, small word clusters tend to have a relatively high similarity with WordNet, while large clusters score lower.

Figure 8 shows that the average similarity values are slightly higher for the selected fifty clusters than for the whole set. Some of this deviation is probably caused by the loss of precision that occurs when the size ranges are selected, since they can never be uniformly distributed.\footnote{16}{In the ideal case, the size ranges would constitute equally large portions of the data.}

5.2 Manual evaluation

In the second evaluation step, we judge the clusters intrinsically\footnote{17}{That is, each cluster is evaluated by its own quality, rather than having external factors affecting the score.} and with our human notions of sense. We have used the same size ranges here. Overall, the scores are higher in this evaluation step, as figure 9 illustrates.
There is a tendency towards better semantic consistency in the smaller clusters. Still, many of the larger clusters hold together surprisingly well.

## 5.3 Clusters compared to WordNet

The following examples are all from the selection of fifty clusters, unless noted otherwise. Average similarity $s_{1,2,3}$ are to the right of each synset.

**Cluster:** \{cufic, oddball\}
**WordNet:** -
None of the words exist as adjectives in WordNet, hence this cluster was not included in the evaluation set.

**Cluster:** \{misbelieving, miscreant, heterodox, heretical\}
**WordNet:** \{dissident, heretical, heterodox\} $0.816$ $0.816$ $0.577$
This case illustrates how easily the score departs from the precious 1.0.

**Cluster:** \{frigorific, refrigeratory\}
**WordNet:** \{frigorific\} $1.0$ $1.0$ $0.707$
This example illustrates the effects of having several measurements; refrigeratory is not in WordNet, so the cluster and the synset become identical once it is removed from the cluster.

**Cluster:** \{suffocating, stifling, smothery, sweltering\}
**WordNet:** \{s weltering, sweltry\} $0.577$ $0.408$ $0.354$
**WordNet:** \{smothering, suffocating, suffocative\} $0.577$ $0.333$ $0.289$
WordNet is more specific in its definitions here, while the generated cluster covers both senses. However, these words could still be interchangeable in many contexts. The cluster can be considered a supersense of the two WN synsets.

**Cluster:** \{drunken, boozy, crapulous, potatory, Bacchic, full, drunk, inebriated, inebriate\}
**WordNet:** \{intoxicated, drunk, inebriated\} $0.436$ $0.436$ $0.385$

**Cluster:** \{inevitable, everyday, mundane, familiar, accustomed, wonted, middling, average, run-of-the-mill, ordinary, white-bread, habitual, routine, second-rate, indifferent, moderate, standard, customary, vanilla, mediocre, straight, regular, bog-standard\}
**WordNet:** \{accustomed, customary, habitual, wonted\} $0.426$ $0.426$ $0.417$
This cluster may be a bit overgrown, still there are some words that are rated as synonyms in the manual evaluation.
6 Discussion

The results point in the direction of the intended result. We have been able to generate good semantic clusters by unifying clusters from the graph-based semantic mirroring application, in accordance with our hypothesis. There is, however, still room for improvement in several aspects.

Word clusters of smaller sizes are usually better than larger ones. Large word clusters tend to spring from large vectors, rather than many smaller ones. Some vectors carry multiple senses from the input data, and can therefore never provide qualities for improvement – they can even be destructive. Some clusters with \( N \) multiple senses are successfully split into \( N \) meaningful clusters when K-means is applied on it again with \( K = N \). For other clusters, however, this instead creates semantically overlapping clusters. The average word cluster size among clusters generated using GBSM is much higher than that of WordNet’s synsets. Manual evaluation turned out to be harder than what was first expected. One can often identify small features, or nuances, of words that disqualify synonymy. Due to the nature of translation, this method is to some extent insensitive to varying levels of specificity. Very specific words may share clusters with less specific ones.

It would be interesting to further investigate the possibility to deterministically apply K-means a third time on some clusters, since manually doing so has proven useful in many cases, but destructive in others. Improvements to the input data from the GBSM application would be another way to enhance results, by for example having a more dynamic setting of when to split a graph compared to the current implementation. Furthermore, deriving the value \( K \) from factors other than the number of unique words when clustering a word cluster space may improve the result, whether or not one uses selective re-clustering.

7 Conclusion

We have found a method of clustering lexical synsets into more useful word clusters. It should be pointed out that the input data needs not originate in a GBSM application at all. One could for example use the Python program presented here to unify thesauruses or other semantic resources created elsewhere. Looking forward, the area of application determines which modificational steps should be taken. The biggest changes are probably attained by modifying the input data, which in this case means tuning parameters in the GBSM application, and/or using additional resources. Tweaking the GBSM application a bit should make way for more concise input vectors, giving a final result that scores higher by all metrics.

Acknowledgements

This work has been funded by the Swedish Research Council (Vetenskapsrådet).

---

18These are not directly comparable, however; WordNet contains many synsets of size 1, while here, such small clusters are discarded before the vector space is initialized. Moreover, WordNet is relatively fine-grained in its definitions.
References


