Extraction of word senses from bilingual resources using graph-based semantic mirroring

by

Hampus Lilliehöök

LIU-IDA/LITH-EX-G--13/008--SE

2013-05-02
Final Thesis

Extraction of word senses from bilingual resources using graph-based semantic mirroring

by

Hampus Lilliehöök

LIU-IDA/LITH-EX-G--13/008--SE

2013-05-02

Supervisor: Magnus Merkel
Examiner: Lars Ahrenberg
Abstract

In this thesis we retrieve semantic information that exists implicitly in bilingual data. We gather input data by repeatedly applying the semantic mirroring procedure. The data is then represented by vectors in a large vector space.

A resource of synonym clusters is then constructed by performing K-means centroid-based clustering on the vectors. We evaluate the result manually – using dictionaries – and against WordNet, and discuss prospects and applications of this method.

Keywords: computational linguistics, natural language processing, data mining, word sense discrimination, semantic mirroring, vector space modeling, cluster analysis
4.3 Manual evaluation ........................................... 32
4.4 Annotated examples ........................................ 33
  4.4.1 Clusters compared to WordNet ................. 33
  4.4.2 Manual evaluation ................................. 34
  4.4.3 Clusters as vectors .............................. 35

5 Discussion .................................................. 39
  5.1 Introduction ............................................ 39
  5.2 Observations ........................................... 39
  5.3 Prospects ............................................... 40
    5.3.1 Selective re-clustering ......................... 40
    5.3.2 Using alternate input data .................... 40
    5.3.3 Changing the function k() ................. 41
  5.4 Conclusion ............................................ 41
Chapter 1

Introduction and background

"Meaning lies as much in the mind of the reader as in the Haiku."
– Douglas R. Hofstadter

The main idea behind this project is to group words by their meanings, based on a method developed by Helge Dyvik [2004]. If two words have the same meaning they are regarded as synonyms. A meaning, or word sense, cannot be seen by looking at a word by itself, it appears when the scope of perspective is widened. It is revealed by context and through interpretation, yet it remains invisible. Although the meaning can neither be touched upon nor fully described, it is designated by symbols – literal strings, or words, in this case.

Word sense clustering can be viewed as the act of identifying relations between word in some resource, and grouping them together according to these relations.

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 thesis are those done by Dyvik [2004] and Eldén, Merkel, Ahrenberg, and Fagerlund [2013], where the latter, although based on the former, lays the foundation to this work – in theory and in practice.

Eldén, Merkel, Ahrenberg, and Fagerlund [2013] developed an application in MATLAB® for carrying out the 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

¹From here on referred to as GBSTM.
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 paths \(^2\) 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 \(^3\), these clusters represent different word senses – often, but not always related to that of the seed word.

### 1.1 The problem

The method has shown promising results, but the effects of combining the results from running the application using several seed words have not yet 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, although commonly, but not always semantically related to the clusters that are obtained, is discarded by default. We want to catch this word as well. Secondly, it is only when the procedure is repeated that a word – as in a sequence of letters – gets a chance to associate with several senses.

For example, the word *green* has these, and possible more meanings, defined here by synonyms:

1. leaf-colored, verdant, lush
2. ecological, environment-friendly
3. new, inexperienced, naïve

Each time the application is executed, any word (except the seed word) will appear in at most one cluster. If the application is executed again using one of the synonymous words above as seed word, *green* is likely to appear in a cluster similar to one above. Applying the procedure repeatedly will thus yield several sets of word clusters that are similar, but not equal, and the clusters will contain seed words of other seed words’ executions. The repetition aims at exhaustingly collecting these semantic relations that otherwise would have been missed, and will by assumption also have the effect of evening out anomalies that may appear. A rich, fine-grained resource for translation data is to prefer in this regard.

\(^2\)A translation path links together two words in some language via a shared translation word in another language.

\(^3\)See Eldén, Merkel, Ahrenberg, and Fagerlund [2013] or Dyvik [2004] for more details.
1.2 Hypothesis and goal

The purpose of this work is to address the issues pointed out, and to extract the underlying semantic structure of the bilingual resource. The procedure of generating word sense data in this manner bears some similarities with data mining; Witten et al. [2011] say the following, regarding large datasets and data mining in general: “Lying hidden in all this data is information – potentially useful information – that is rarely made explicit or taken advantage of.” Firstly, in the experiment carried out by Eldén et al. [2013], implicit semantic data, generated from many hours of manual lexicographic labor, is extracted. Secondly, in this work, the unified word sense resource is something that is assumed to lie hidden in the sets of clusters generated by executing GBSM for each seed word in the dictionary.

1.3 Process

To do this we extend the GBSM application used by Eldén, Merkel, Ahrenberg, and Fagerlund [2013] to generate the word clusters for all available seed words, and unify the result in a data-driven clustering operation 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:

1. Manually: by examining a randomly selected subset of the output data, with the aid of linguistically educated people – in this case my supervisor.


1.4 Report structure

As for the structure of the report, the next chapter describes the various components that have been used. Both technical components and theoretical concepts are regarded as such, and they are presented in an order as to give answers to the questions “why?” and “how?”, starting with linguistic concepts and terminology. In chapter 3, the practical parts – or implementations of the components – are explained in greater detail, and we follow the input data step-by-step on its path through them. The results obtained are presented in chapter 4, while chapter 5 provides a general discussion and the conclusions.
Chapter 2
Components and sources

In this chapter the various components and sources are described as well as some theoretical base concepts regarding linguistics and data clustering.

2.1 Words, lemmas and lexemes

An important distinction here is the separation of words from their meanings, but also what constitutes a word. A word as it occurs in a text may be viewed as a sequence of orthographic units (letters), a composition of morphemes, a syntactic unit or a symbol denoting one of several meanings – or word senses.

A term that is commonly meant by “word” is lemma. A lemma is the unconjugated form, commonly found in a lexicon, of a lexeme, i.e. a set of word forms – representing one or several meanings. For example, for nouns, “bike” is a lemma while “bikes” is not, since it is an alternate word form (plural) of the same lexeme. “Biker” on the other hand, although it is derived from the same word stem, represents another lexeme and another meaning, and is thus a lemma. For verbs, “(to) look” is a lemma, as opposed to other word forms of the lexeme like “looked”, “looks” and “looking”. “Looks” is however also the lemma of the (special plural form) noun-lexeme near-synonymous to “appearance”.

In this report, and in the data described, English adjectives are used, hence there exist no conjugations for grammatical agreement regarding gender, person, numbers etc.

2.2 Word senses and semantic relations

Jurafsky and Martin [2009] use the term lexeme to denote a pair with a semantic meaning on one hand, and a set of word forms on the other. There may exist several identical lexemes in a lexicon. Homonymous lexemes are identical in their set of word forms but different in their meaning. These are usually found in a lexicon as word\(_1\), word\(_2\), etc. However, some set of closely related word senses may be denoted by a single lexeme. In this case, the lexeme is said to be
polysemous, or polysemy is said to apply between the related word senses. A common example is the word *bank*, whose meanings (roughly) “financial institute” and “sloping mound” are said to be homonymous, while a financial bank and a blood bank are said to be polysemous [Jurafsky and Martin, 2009]. It is, however, hard to draw a line between homonymy and polysemy and that makes it hard to decide how many word senses there are. Dictionaries often disagree on this.

![Figure 2.1: A polysemous relationship between word senses.](image)

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. That being the case, no two words can be absolute synonyms [Jurafsky and Martin, 2009].
It should also be pointed out that word senses may be defined using various degrees of strictness, i.e. semantic features in various distance from the word sense’s semantic core are included or excluded. Consider figure 2.4. Some definitions may include the whole semantic subspace (marked with dots), and even other word senses in their entirety, while others include only the features that constitute Word sense 1. Definitions are those of lexicographers, speakers of the language and everyone else who may have an opinion.

Jurafsky and Martin [2009] claim that dictionaries may be less generous in this aspect (i.e. they are more fine-grained and contain more word senses), while for computational applications the senses may be kept larger and fewer.
2.3 Semantic mirroring, linguistic motivation 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. ¹ 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, courtesy of Eldén, Merkel, Ahrenberg, and Fagerlund [2013], to give some idea of how the method can be applied. The example is POS-insensitive ², but is in other aspects similar to how it has been used in the GBSM application:

The Swedish seed word rätt is looked up in a dictionary, and the following

---

¹An aligned parallel corpus is a set of sentences (or more arbitrary sequences of words) in one language paired with their translations. This text will however focus on single words and their translations, i.e. dictionaries, as far as semantic mirroring is concerned.

²POS, or part of speech, also known as word class, denotes different classifications of words based on their syntactic and morphologic behavior. The most well-known examples are adjectives (e.g. heavy), nouns (e.g. aircraft) and verbs (e.g. (to) fly).
translations are found:

\{course, dish, a lot, meal, proper, right, justice, rightly, somewhat, correct, directly, fair and square, full, law, court, quite, fairly\}.

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, lärkurs, 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. \(^3\) 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. Eldén, Merkel, Ahrenberg, and Fagerlund \cite{2013} developed a program in MATLAB\textsuperscript{®} for carrying out the procedure. The translation matrix was generated from a subset of Norstedts Swedish – English lexicon, containing only adjectives \cite{Norstedts, 2000}.

As with most algorithms, there is no strict definition of how semantic mirroring is done, and thus the procedure may be configured in several ways. The selection of source affects what translations and translation paths there are to begin with, and hence what data that can be extracted. No two dictionaries, thesauruses \(^4\) or parallel corpora are identical.

### 2.4 Python

Python is a high-level interpreted programming language. Python has a clear, natural language-like syntax, and operations on files and text strings can commonly be expressed in few lines of code. Further on, it runs on many platforms, and a multitude of extension modules and packages are available. In this project Python was primarily chosen for two reasons: its ease of use with text data, and the availability of the NumPy package.

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

\(^4\)A thesaurus is a dictionary of synonyms.
In [109]: import numpy as np
In [110]: a = np.random.permutation(100)
In [111]: a
Out[111]: array([39, 98, 37, ..., 62, 26, 5])
In [112]: %timeit sum(map(lambda x: x*x, a)) ** 0.5
   10000 loops, best of 3: 151 us per loop
In [113]: %timeit np.linalg.norm(a)
   100000 loops, best of 3: 16.6 us per loop
In [114]: %timeit np.dot(a, a) ** 0.5
   100000 loops, best of 3: 11.4 us per loop
In [115]: %timeit np.sqrt(np.dot(a, a))
   1000000 loops, best of 3: 6.31 us per loop
In [116]: sum(map(lambda x: x*x, a)) ** 0.5 == np.linalg.norm(a)
   == np.dot(a, a) ** 0.5 == np.sqrt(np.dot(a, a))
Out[116]: True

Figure 2.5: A vector norm is calculated in four different ways to illustrate performance.

2.4.1 iPython

iPython [Pérez and Granger, 2007] is a powerful interactive Python shell. It offers a practically seamless integration with the operative system and features such as clipboard pasting, performance measuring and quick documentation/API access.

2.4.2 NumPy/SciPy

NumPy is a package for Python, presented as an open-source alternative to MATLAB®. It is based on earlier mathematical Python tools, but most notably unified into one resource by Travis E. Oliphant. Its perhaps most significant feature is the multidimensional array (similar to the MATLAB® matrix). NumPy is part of the SciPy library [Jones et al., 2001–].

NumPy increases the performance of mathematical operations, especially those operating on large arrays (or vectors/matrices). NumPy has been of great help in this project for several reasons: A vector space model is the central component of the program, thus much functionality needs not be rewritten, and it makes way for more readable code. Also, the vectors are very large, the operations needed to be done on them are often done on the vectors as a whole (as opposed to elementwise), and the operations needed to be done are numerous.

Figure 2.5 shows a quick performance demonstration; the norm (or length) of a vector of 100 random elements is calculated in different ways using the following methods respectively: Python built-ins only; NumPy’s predefined norm function; NumPy’s dot product along with Python’s built-in power operator; NumPy’s dot product along with NumPy’s square root function. The fastest way is almost 24 times faster than the slowest. Performance measurement is courtesy of iPython.
Another library included under the SciPy umbrella is Matplotlib. The package allows for comprehensive graph plotting and has MATLAB-like syntax and features. In this project, Matplotlib has been used for plotting figures 2.7, 3.7, 3.8, 3.9, 4.2 and 4.3.

2.5 WordNet and NLTK

WordNet [Miller, 1995] is a large network of concepts, or meanings, and their associated English words. The WordNet introduction says the following: “WordNet® is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations.”

WordNet has been an on-going project at Princeton University since the mid-80’s. It acts as a dictionary, thesaurus and a hierarchical knowledge representation system. It is freely available for anyone, as interactive use or file download. WordNet has been cited many times in the field of computational linguistics and others, and is often used as a gold standard for evaluation of automated extraction of similar data (Cicurel et al. [2006], Bansal et al. [2012]).

The NLTK, or Natural Language Toolkit, package [Bird et al., 2009] is a collection of language data and tools that can be imported in Python programs. NLTK is free and open-source. Among its many features, NLTK offers a WordNet interface, which has made the programming part of this project much easier.

2.6 Clustering

“Cluster analysis or clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) 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

---

5http://wordnet.princeton.edu/wordnet/
6http://en.wikipedia.org/wiki/cluster_analysis
Figure 2.7: A simple clustering example.

et al., 2011]. Figure 2.7 is a graph of points generated around two points in 2-dimensional space. The coloring indicate how one would (probably) intuitively group, or cluster, these points into two sets. More importantly, it can also be done algorithmically. Some general ways of doing so are described below.

Clustering algorithms exist in many flavors, and are configurable in many ways. They may differ in various aspects depending on the scenario, most importantly in how their start states are defined; in what order cluster association data is generated; how similarity between data items is measured; and which data items that are involved when the similarities are measured.

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.

---

7One configuration of this algorithm is commonly known as $K$-means or Lloyd’s algorithm.
Chapter 3

Method

In this chapter, the practical work done is described, following the procedural steps of the input data.

3.1 Extending the GBSM application

The first practical step in this work has been to gather all the required data. As mentioned earlier, the MATLAB® application developed by Eldén, Merkel, Ahrenberg, and Fagerlund [2013] for their experiment have been used for this. The explanation in the introduction, along with remarks here, is hopefully sufficient for comprehending what has been changed or added (see chapter 1). For a more detailed description of this procedure, however, please see [Eldén, Merkel, Ahrenberg, and Fagerlund, 2013].

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, in agreement with my supervisor.

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. In effect, this means that 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.

Lastly, modifications for writing the cluster data to file have been done, as well as the inclusion of exception handling. (Some seed words caused errors that were also apparent in the original edition of the application – these could not be used for generating semantic clusters.)

Output data are all clusters of English adjectives, derived from graphs with
three or more nodes \(^1\) associated with the seed words that were used to retrieve them.

### 3.2 Python programming

Python enters the picture where the GBSM procedure ends. Originally a script for deriving statistics from the GBSM application’s output data, the application grew to become the clustering application used for the remainder of the project.

#### 3.2.1 Dataset

As mentioned earlier, the dataset, or input data \(^2\), are clusters of English adjectives, some of which can be seen in figure 3.1. Each section separated by `##` represents the result of one iteration in the main loop of the modified MATLAB\(^\text{®}\) application. The first two items of numerical data are irrelevant to this report. 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.

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

---

\(^1\)Nodes are words in a common language, that share some translation path(s).

\(^2\)We’re entering a new phase and the output data of the GBSM application is henceforth referred to as *input data* or *dataset* to avoid confusion.
Figure 3.2: The purpose of section 3.1 – a set of possibly overlapping word clusters is yielded from a set of seed words.

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

Figure 3.3: The purpose of section 3.2 – the word clusters from section 3.1 are clustered into sets of word clusters. The unique word set of the clustered clusters are considered the result.

To emphasize and conclude, although this may be a bit confusing at first: 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 3.2 and 3.3).

In contexts where disambiguation is difficult, most notably in chapter 4, the term GBSM output cluster refers to a word cluster, while the word cluster denotes output from the K-means algorithm.

3.2.2 Reductions in the dataset

Reductions in the dataset have been made, in order to both decrease its size and increase its quality. In agreement with my supervisor, 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 semantic properties. Clusters of sizes above 200 are not uncommon. By comparison, the largest synset of adjectives in WordNet (see figure 2.6) is of size 23, and the average size is \(\sim1.65\).
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 3.2.5). 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.

Further on, we have decided to discard word clusters of size 1 within the clustering procedure. There are justifications for this too – if a cluster contains only one item, one of two scenarios apply:

1. The cluster is orthogonal to all other clusters (see section 3.2.3 for how this works)
2. The cluster is a subset of one or more other clusters

In the first case, the cluster would share no similarities with other clusters (i.e. cosine of their angle in the vector space would be 0), and there would not be any reason to group them together. Being isolated in this manner, these clusters neither affect, nor are affected by other clusters – hence they do not contribute to the semantic data extraction. In the second case, the justification is exactly the same as for removing duplicate clusters.
Figure 3.6: The dataset after the final reduction. Clusters of sizes 2-15 are allowed, not including duplicates.

The data in figure 3.6 is the data that will be used for the remainder of the application. Of course, this heavily reduces the number of words that are output from the program as a whole.

The following figures show how allowing different cluster sizes affects some properties (see the y-axis of each graph) of the input data.

Figure 3.7: The number of clusters in the dataset as the upper threshold is increased.
**Figure 3.8:** The number of unique words in the dataset as the upper threshold is increased.

**Figure 3.9:** The average cluster size in the dataset as the upper threshold is increased.
3.2.3 Vector space model

The input data that was not discarded constitute the central resource for the remainder of the application. NumPy (section 2.4.2) has been used for this.

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.

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

Using the reduced input data (figure 3.6): \(n = 8832\), \(m = 13691\). The graphic style of the matrix bears some resemblance to the data structures used for its representation. NumPy’s fundamental component – the numeric array – made this very convenient. 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.

3.2.4 Partitioning of vector space

As mentioned in section 2.6, clustering in general is computationally hard in large and/or high-dimensional datasets. Depending on the configuration (see section 3.2.1) – 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 3.2.3), 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 of this, \( M \) can be reduced into the two matrices \( M'_1 \) and \( M'_2 \):

\[
M'_1 = \begin{pmatrix}
\begin{bmatrix}
a & b & c & d & e \\
u & 1 & 0 & 0 & 0 \\
v & 1 & 1 & 0 & 0 \\
x & 0 & 0 & 1 & 1 \\
y & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\end{pmatrix}
\]

\[
M'_2 = \begin{pmatrix}
\begin{bmatrix}
a & b & c & d & e \\
w & 0 & 0 & 1 & 1 \\
x & 0 & 0 & 1 & 1 \\
y & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\end{pmatrix}
\]

Or even simpler:

\[
M''_1 = \begin{pmatrix}
\begin{bmatrix}
a & b \\
u & 1 \\
v & 1 \\
\end{bmatrix}
\end{pmatrix}
\]

\[
M''_2 = \begin{pmatrix}
\begin{bmatrix}
c & d & e \\
w & 1 & 1 \0 \\
x & 0 & 1 & 1 \\
y & 0 & 0 & 1 \\
\end{bmatrix}
\end{pmatrix}
\]

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. Thus, this turned out to be only a minor reduction of the critical factors compared to the prior steps (see section 3.2.2).

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; were the vector space not partitioned, the clustering algorithm could in some cases join the dissimilar vectors with others. Additionally, since cosine similarity is used as the

---

3 Explained in section 3.2.5.
4 Some unfortunate properties of the result have been traced to properties of the K-means algorithm (section 3.2.5). This is discussed in chapter 5.
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.

3.2.5 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. Like the rest of the program, it has been implemented in Python, and uses the NumPy-based vector space described in section 3.2.3.

As mentioned in section 2.6 regarding clustering, each category of methods has its drawbacks. 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.

K-means was chosen because of good available documentation – Witten et al. [2011] provided a good introduction to the algorithm – and the close correspondence between the value \( K \) and the expected number of data items (word clusters), and therefore words, per cluster.  

**Description and formal definition**

Following the introduction to centroid-based clustering given in section 2.6, below is a description of the algorithm as implemented in this application. It follows the pseudocode in Algorithm 1.

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 using Algorithm 2 and stored in \( C \). 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 Algorithm 3, 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.

---

5This, however, turned out to be not entirely unproblematic, see chapter 5.

6Note that the centroid vector is squared, the reason for this is discussed in chapter 5.
Algorithm 1 K-means – partition $D$ into most similar subsets

$D \leftarrow$ all data items, or vectors
$N \leftarrow \text{size}(D)$
$K \leftarrow \text{int}, \; K \leq N$
$C \leftarrow \text{array}(), \; \text{size}(C) = K$
$D_1, D_2, \ldots, D_K \subset D, \; \bigcap_{i=1}^{K} D_i = \emptyset, \; D \setminus \bigcup_{i=1}^{K} D_i = \emptyset$
repeat
  for $i = 1 \rightarrow K$ do
    if not empty($D_i$) then
      $C[i] \leftarrow \text{centroid}(D_i))^2$  \(\triangleright\) weighted towards more frequent items
    end if
  end for
  no_of_operations $\leftarrow 0$
  for $i = 1 \rightarrow K$ do
    for item $\in D_i$ do
      nearest_centroid $\leftarrow C[i]$
      nearest_centroid_metric $\leftarrow \text{metric}(C[i], \text{item})$
      changed $\leftarrow 0$
      for centroid $\in C$ do
        metric $\leftarrow \text{metric}(\text{centroid}, \text{item})$
        if metric $> \text{nearest_centroid_metric}$ then
          nearest_centroid $\leftarrow \text{centroid}$
          nearest_centroid_metric $\leftarrow \text{metric}$
          changed $\leftarrow 1$
        end if
      end for
      if changed then
        $D_{\text{index}(\text{nearest_centroid})} \leftarrow D_{\text{index}(\text{nearest_centroid})} \cup \{\text{item}\}$
        $D_i \leftarrow D_i \setminus \{\text{item}\}$
        no_of_operations $\leftarrow \text{no_of_operations} + 1$
      end if
    end for
  end for
until no_of_operations $= 0$

Algorithm 2 Centroid – calculate centroid vector for vectors

$k \leftarrow \text{size}(\text{vectors})$
$\text{sum\_vectors} \leftarrow \sum_{i=1}^{k} \text{vectors}[i]$
$\text{sum\_lengths} \leftarrow \sum_{i=1}^{k} \|\text{vectors}[i]\|$
return $\frac{\text{sum\_vectors}}{\text{sum\_lengths}}$
**Algorithm 3** Metric – compare vectors \( v, w \)

\[
\text{return } \frac{v \cdot w}{\|v\| \|w\|} \quad \triangleright \text{real value between 0 and 1}
\]

**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 \( K \)
3. The number of iterations \( I \) needed to reach a solution. \(^7\)

The computational time complexity of the algorithm is in \( O(N \times K \times I) \). 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. Fortunately the NumPy library is well-suited for such applications (see figure 2.5 for a comparison of alternatives).

**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. As mentioned in section 3.2.4, 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 from section 3.2.4 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. \(^8\) Next, K-means was applied on each subspace, with \( K \) derived from the number of unique words in that subspace. \(^9\) Figure 3.10 gives an illustration of this.

What was then obtained is presented in chapter 4.

---

\(^7\)Hard to predict, but depends on \( N, K \), the type of data and the metric function. To give some hint, this number has been between 5 and 30, depending on the configuration.

\(^8\) \( \frac{N}{K} = \frac{13085}{15} \approx 872.3 \)

\(^9\) \( K_{\text{subspace}} = \frac{\text{number of unique words in subspace}}{\text{average cluster size in vector space}} \)
3.3 Evaluation

3.3.1 WordNet as gold standard

WordNet [Miller, 1995] has a long history, a good reputation and is widely regarded as a gold standard in related academic fields.

Remembering the discussion in 2.2, one may conclude that no universal truth exists for the general problem of defining and delimiting word senses. Thus, the main drawback of using WordNet for evaluation is that its definitions are accepted as truth, where in fact no truth really exists. However, WordNet is an extensive resource, performing well at approximating truth, 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 are selected at random on five intervals of cluster sizes, aiming at an even distribution. (I.e. ten clusters are selected for each interval.)

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. 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). As a parenthesis, one should bear in mind that intactness is sometimes to prefer over WordNet-similarity.

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.

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

Figure 3.11: 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 both 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.

### 3.3.2 Manual evaluation

As a second step in the evaluation, the two people involved in this project – me and my supervisor – evaluate the set of fifty word clusters manually. The

---

10 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?” When approximating a gold standard, one is more likely to succeed if one has similar data.

11 A similarity \( \in (0, 1], \in \mathbb{R} \) is obtained, where 1 means that the compared sets of words are identical.
clusters are the same as in the previous step. In this step, we rely primarily on our “gut-feeling”, but use dictionaries and example sentences for verification. We examine the clusters independently of each other, and for each cluster, we identify the predominant sense, and then calculate a precision score using the formula

\[
\frac{\text{number of words satisfying the sense}}{\text{number of words in cluster}}
\]

This will be equal to 1 in the best cases, and approach 0 in the worst.

Using this metric, we compare and discuss our individual interpretations and scores, and reach conclusions by consensus, or by calculating the average score in the cases where we disagree.

3.3.3 Annotated examples

Additionally, for a smaller amount of word clusters, we present their actual contents, along with their associated WordNet synsets and some annotations. We also give some illustrations of how word clusters are made from combining vectors, and discuss phenomena this arise from this.

3.3.4 A clarification on metric scores

Although one may be tempted to compare the results of the two scores against each other, one must bear in mind that they measure two different properties. A high WordNet similarity means exactly that; a word cluster is equal or very similar to a WordNet synset. Semantically it implies that the method used has to some extent simulated the semantic delineation once done by Miller [1995]. A high score in the manual evaluation, on the other hand, means that the synset has satisfied our human notion of sense, and that the dictionaries used have given evidence for it. Indeed, a score close to one is to prefer in both scores, but they still mean two different things.
Chapter 4

Results

In this chapter we present some output data, i.e. clusters generated by the K-means algorithm. Results from our evaluation of the semantic qualities of these clusters are also presented here, along with some general statistics regarding the output data set.

4.1 Output data and its distribution

Out of the 13691 output clusters from the GBSM application, 2727 clusters of clusters, or senses, were distinguished by K-means. Figures 4.1 and 4.2 show how these are distributed by the criterion number of unique words. Figure 4.3 shows how the senses are distributed by the criterion number of constituent vectors. The distributions are similar, and as one may suspect, larger word clusters tend to consist of many vectors, while smaller ones tend to consist of fewer vectors. 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 4.1: Distribution of clusters by their number of unique words. The ranges were chosen to make the groups as equal in size as possible.
**Figure 4.2:** Distribution of clusters by their number of unique words. The data is the same as in figure 4.1, only displayed differently. There are no clusters of size 1, yet smaller clusters are clearly more common than large ones.

**Figure 4.3:** Distribution of clusters by their number of vectors. Although this plays no formal role in the evaluation, it gives some insight about the output data. Note that the distribution is similar in shape as the one in figure 4.2.
4.2 WordNet as gold standard

Below is shown the result of the first evaluation procedure. Average similarity$^{1,2,3}$ are the three different measurements described in section 3.3.1. Since we suspected that clusters of different sizes would not score equally well, scores for the size ranges introduced in section 4.1 have been measured, in addition to the overall score.

<table>
<thead>
<tr>
<th>Cluster size</th>
<th>Avg. sim.$^1$</th>
<th>Avg. sim.$^2$</th>
<th>Avg. sim.$^3$</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 4.4: Average similarity with nearest WordNet synsets.

<table>
<thead>
<tr>
<th>Cluster size</th>
<th>Avg. sim.$^1$</th>
<th>Avg. sim.$^2$</th>
<th>Avg. sim.$^3$</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 4.5: Average similarity with nearest WordNet synsets.

As figure 4.4 suggests, small word clusters tend to have a relatively high similarity with WordNet, while large clusters score lower.

Figure 4.5 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.$^1$

As we shall see in chapter 5, we have ideas about what parameters to adjust in order to obtain smaller clusters that score higher in this test.

4.3 Manual evaluation

In the second evaluation step, we judge the clusters intrinsically$^2$ 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 4.6 illustrates.

$^1$In the ideal case, the size ranges would constitute equally large portions of the data, see figure 4.1.

$^2$That is, each cluster is evaluated by its own quality, rather than having external factors affecting the score.
Cluster size | Avg. score
---|---
2-4 | 0.975
5-7 | 0.742
8-11 | 0.764
12-17 | 0.807
18-70 | 0.725
all | 0.802

*Figure 4.6:* Average scores for manually evaluated clusters.

There is a tendency towards better semantic consistency in the smaller clusters. Still, many of the larger clusters hold together surprisingly well.

### 4.4 Annotated examples

#### 4.4.1 Clusters compared to WordNet

The following examples are all from the selection of fifty clusters, unless noted otherwise. We aim to illustrate various circumstances affecting the score in one way or another. Average similarity $\text{1,2,3}$ are to the right of each synset.

**Cluster:** \{agglomerate, conglomerate\}

**WordNet:** \{agglomerate, agglomerated, agglomerative, clustered\} $0.707$ $0.354$

**WordNet:** \{conglomerate\} $0.707$ $0.707$ $0.707$

WordNet has two synsets containing at least one of the words in the cluster. Since there are no candidates for removal, the two metrics for each synset have the same value.

**Cluster:** \{cufic, oddball\}

**WordNet:** -

None of the words exist as adjectives in WordNet, hence this cluster was not included in the evaluation set. We shall have more to say about this cluster in section 4.4.3.

**Cluster:** \{misbelieving, miscreant, heterodox, heretical\}

**WordNet:** \{dissident, heretical, heterodox\} $0.816$ $0.816$ $0.577$

A fine cluster in our opinion. 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 two measurements; *refrigeratory* is not in WordNet, so the cluster and the synset become identical once it is removed from the cluster. In a real-life scenario, however, we probably would not
want to remove *refrigeratory*.

**Cluster:** \{suffocating, stifling, smothery, sweltering\}

**WordNet:** \{sweltering, 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 our cluster covers both senses. However, these words could still be interchangeable in many contexts. The cluster can be considered a supersense of the two synsets.

**Cluster:** \{two-tongued, double-tongued, lying, mendacious\}

**WordNet:** \{ambidextrous, deceitful, double-dealing, duplicitous, Janus-faced, two-faced, double-faced, double-tongued\} 0.25 0.177

**WordNet:** \{mendacious\} 0.707 0.707 0.5

Again, our sense space is a bit wider than WordNet’s. WordNet defines the first synset as “marked by deliberate deceptiveness especially by pretending one set of feelings and acting under the influence of another”, and the other as “given to lying”. One can see how *deceitful* would match both descriptions, and one may wonder why *lying* isn’t part of any of the sets. (Participles are otherwise common as adjectives in WordNet.)

**Cluster:** \{drunken, boozy, crapulous, potatory, Bacchic, full, drunk, inebriated, inebriate\}

**WordNet:** \{intoxicated, drunk, inebriated\} 0.436 0.436 0.385

One of the benefits of doing this is that you get to learn lots of new words. This example also illustrates how adjectives can be morphologically constructed from verbs in several ways. In the first example in this section, WordNet contains both *agglomerate* and *agglomerated*, whereas here only the participle form *inebriated* is present.

**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 synonyms in our opinion, but not according to WordNet; *routine* and *regular*.

**Cluster:** \{Scottish, Scotch, Scots\}

**WordNet:** \{Scots, Scottish, Scotch\} 1.0 1.0 1.0

This cluster is not from the selected fifty, but we wanted to show that some WordNet synsets are re-created by our implementation.

### 4.4.2 Manual evaluation

To give some idea about how we evaluated the clusters, a few examples and their scores are presented here. The scores are set as described in section 3.3.2.
Cluster: \{heretical, heterodox, misbelieving, miscreant\} 4/4
A satisfactory cluster in our opinion – the words are interchangeable in many contexts.

Cluster: \{smothery, stifling, suffocating, sweltering\} 4/4
By the same reasoning, this is also good.

Cluster: \{magisterial, pontifical, professorial, sententious, didactic\} 3/5
We agreed that while this cluster clearly points in a direction, its semantic features are too wide-spread to receive a full precision score. In our view the three highlighted words have a religious or moral sense, while didactic and professorial (and perhaps also magisterial) lean towards academic teaching.

4.4.3 Clusters as vectors
Here we highlight the word clusters’ being composed by vectors, with some comments on related phenomena.

\[[\text{bottommost}, \text{nethermost}, \text{lowermost}]\] :
- [bottommost, nethermost]
- [bottommost, lowermost]
- [nethermost, lowermost]

\[[\text{Scottish}, \text{Scotch}, \text{Scots}]\] :
- [Scottish, Scotch]
- [Scotch, Scots]
- [Scottish, Scots]

These two clusters illustrate the main idea of combining the results from multiple executions of GBSM – meaning is extracted through combining the results. They both have identical synsets in WordNet.

\[[\text{Cufic}, \text{oddball}]\] :
- [Cufic, oddball]

This single-vector cluster appears already in the GBSM data. Since the words do not occur elsewhere in the input data, it is a disjoint cluster space (as described in section 3.2.4) and was therefore never combined with any other vector. Neither can it be decomposed any further.

\[[\text{Hyperborean}, \text{arctic}, \text{northern}, \text{boreal}, \text{northward}, \text{north}, \text{north-polar}]\] :
This is a disjoint subspace, before being clustered by the K-means algorithm. Below is the result of applying K-means once with $K = 2$, as is the case in the final result.

The vectors have been grouped together by similarity, yet in this example the vectors' contents are too intertwined to yield two word clusters with different senses – there is no obvious “cutting point” among the vectors. Using a strict definition, two or more senses are present in both resulting clusters. The trivial case of $K = 1$ would have left the original subspace unchanged, while $K = 3$ would have enhanced the undesired effect of spreading the senses over unnecessarily many clusters. This is shown below.
This illustrates how the behavior of the application depends on the input data. At the current stage, there is no way of discarding vectors containing multiple senses. We can, however, to some extent affect how the GBSM application behaves. In the next chapter we discuss the possibilities and implications of this.

[south, southern, meridional, antarctic, southward, austral] :

[south, southern]
[south, antarctic]
[southern, antarctic]
[meridional, antarctic, southward]
[meridional, antarctic, austral]
[antarctic, southward, austral]
[meridional, southward, austral]

The antonymous cluster is more concise.

[glittering, aglitter, scintillant, ablaze, spangly, brilliant, sparkly, lurid, flamy, flamboyant, blazing, fiery, flaming] :

[glittering, brilliant, sparkly]
[glittering, scintillant]
[glittering, aglitter]
[glittering, aglitter, spangly]
[ablaze, spangly]
[aglitter, ablaze, spangly]
[ablaze, blazing, fiery]
[ablaze, flamy, blazing, fiery, flaming]
[ablaze, lurid, flamy, flamboyant]

This cluster is taken from the final output data. Below is the result of applying K-means a second time, with $K = 2$.

[glittering, aglitter, scintillant, spangly, brilliant, sparkly] :

[glittering, brilliant, sparkly]
[glittering, scintillant]
[glittering, aglitter]
[glittering, aglitter, spangly]

[aglitter, ablaze, spangly]
[ablaze, spangly]
[aglitter, ablaze, spangly]
[ablaze, blazing, fiery]
[ablaze, flamy, blazing, fiery, flaming]
[ablaze, lurid, flamy, flamboyant]

We conclude that some clusters benefit from a second K-means pass, while others do not. This is also discussed in the next chapter.

[assistant, conjunct, adjunct, under, nether, spare, supernumerary, supererogatory] :
This example illustrates how the word *assistant* acts as a bridge between several few-word clusters. More formally, words in this cluster are connected by transitivity.
Chapter 5

Discussion

In this chapter the results are discussed more freely. General observations are listed and explained. Probable causes for problems are identified, and ideas for solutions to these problems are presented. Additionally, prospects for improving the result and extending the features are given.

5.1 Introduction

The results at least point in the direction of the intended result. We have been able to generate good semantic clusters by repeating the semantic mirroring procedure, in accordance with our hypothesis. There is, however, still room for improvement in several aspects. Possible causes for, and outlines for solutions to these problems are discussed below.

5.2 Observations

- Word clusters of smaller sizes are usually better than larger ones.
- Large word clusters tend to spring from large vectors (i.e. large GBSM output clusters), rather than many smaller ones.
- Some vectors carry multiple senses from the GBSM 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 GBSM output clusters is much higher than that of WordNet’s synsets.  

\footnote{These are not directly comparable, however; WordNet contains many synsets of size 1, while here, such small GBSM clusters are discarded before the vector space is initialized. Moreover, WordNet is relatively fine-grained in its definitions.}
• 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. Dictionaries proved to be very useful here. ²

• Squaring the centroid vectors in the K-means algorithm (algorithm 1) favors cases where words occur in multiple vectors when metrics are calculated. This eliminates many, but not all cases of single uncommon words causing clusters to grow very large by transitivity. The result is better and reached in fewer iteration steps.

• Due to the nature of translation, this method is to some extent unsensitive to varying levels of specificity. Very specific words may share clusters with less specific ones.

5.3 Prospects

Some potential changes and additions are discussed. Changing or tuning these will have different effects on the output data, and one probably want to choose one’s strategy depending on where the system is to be implemented, whether the intended result is more WordNet-like or not, etc.

5.3.1 Selective re-clustering

One idea of a continuation is to decide on a method for deterministically applying K-means a third time on some clusters, since manually doing so has proven useful in many cases, but destructive in others. (For example, see the “glittering” cluster in section 4.4.3.)

5.3.2 Using alternate input data

The input data is obviously a major factor in how the result of this procedure turns out. Given the GBSM setup in this scenario, the clustering could be tuned by changing the fiedler value threshold used in the GBSM application and generating new input data. As mentioned earlier, a fiedler value of 0.2 was used when the input data was generated, and we have reason to believe that a slightly smaller fiedler value should entail smaller GBSM output clusters, and therefore also in the final output data. To let this value be determined dynamically by some function could be of interest.

It should also 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.

²http://www.merriam-webster.com/
5.3.3 Changing the function \( k() \)

Deriving the value \( K \) from factors other than the number of unique words when clustering a vector space may improve the result, whether or not one uses selective re-clustering. One factor that may be of interest is the size of the word space’s constituent vectors.

Including more factors would allow for finer-tuned configurations and perhaps better adaption to particular datasets. Since some cluster spaces are successfully decomposed into good semantic word clusters, while other decompositions give rise to semantic overlap among the results, we have reason to believe that the coarse-grained determination of \( K \) is at least partially responsible.

Another idea is to provide \( k() \) with some look-ahead abilities. Vector similarity could be used here as well.

5.4 Conclusion

We may say, with some confidence, that we have found a method that is useful in its current state, and even more so if tweaked a bit. In this chapter, we have discussed the overall qualities of the result, and identified various factors affecting it in one way or another.

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 both metrics. Of course one should be careful not to be blinded by numbers, but we hope to have shed some light on the causal relationships between the various parameters and the result.

For our linguistic assumptions to hold, the input data needs to be on the same form and as semantically correct as possible, but how it is generated really does not matter much. However, we are excited about graph-based semantic mirroring, and we have worked with this method in mind from the start.
Bibliography


På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtidiga ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© [Hampus Lilliehöök]