A method for finding word clusters in spoken language

Leif Grönqvist and Magnus Gunnarsson
{leifg, mgunnar}@ling.gu.se

1. Introduction

At the Department of Linguistics in Gothenburg, a research group led by Jens Allwood has been working with spontaneous spoken language since the late 70's. The authors of this paper are currently a part of this group. We think that it is important to be aware that spoken language is different from written language, and that methods and theories developed for written language are not necessarily suited for spoken language. For a description of the differences between spoken and written language, see e.g. (Allwood 1996).

That words can be divided into groups is a fairly natural idea for anybody who has studied language, and indeed the concept of parts-of-speech goes as far back as the study of language does. But exactly which groups should be used and what their basis should be, is still an unresolved issue. It would be a great advantage if one could find theory-neutral groups of words, which could be used to inspire and verify theories about parts-of-speech. In this paper we try to find similarities between words, and thus to find word groups, in spoken language without using traditional written language theories about grammar and parts-of-speech. Instead we want to induce information directly from our corpus[1] and then evaluate what we have found. We are not aiming at creating general tools for automatic parts-of-speech tagging or semantic clustering or the like.

It should be noted that the grouping of syllables/phonemes into words is not unproblematic, and that our work in this aspect relies on the transcribers' intuition, which can be expected to be heavily influenced by written language. However, we consider that a different problem.

The idea for this paper was born during a collaborative project between the Danish researcher Peter Juel Henrichsen and the research group in Gothenburg, working with the GSLC (Göteborg Spoken Language Corpus) (Allwood et al 2001) and the BySoc corpus (Henrichsen 1997), (Gregersen 1991). As a part of the project we made a variety of comparisons between Swedish and Danish. One of Henrichsen's contributions was a context based measure of similarity between words in a corpus. The context window is extremely small – namely 1+1 word – one to the left and one to the right. The measure is based on the number of mutual context words, and words with a high similarity score are called *siblings* (Henrichsen 2003). If two words have exactly the same relative distribution of context words, both to the left and to the right, the similarity will be 1.0, and if the words do not have any context words in common it will be 0.0.

2. Constructing a new iterative word clustering algorithm

2.1 Using the sibling measure to find word clusters

Henrichsen's purpose with the sibling measure was to find the most similar pairs of words. Quite obviously, in many cases several words will be rather similar, while only two of them will be the closest pair. One way of finding word clusters could be to define a threshold Thr_{score}, and to say that if word A resembles word B more than Thr_{score}, then A and B belong to the same cluster. What is more, since the sibling measure is asymmetrical, 'sibling chains' may occur, where A resembles B, B resembles C, C resembles D etc. This will also help to form clusters.

There are several problems with this approach. Firstly, there is no way of adding more words to the clusters afterwards. Either the words resemble each other enough to pass the threshold, or they do not. In practice, that would mean that the word clusters would not have more than 10-20 members, at most. Secondly, the 'sibling chains' may cause some problems, since there is no obvious way of stopping them – if we are unlucky, the chains can go on until far too many words are included. An example could be homonyms like the Swedish *var* (either a verb, *was/were*, or an adverb, *where*): the word *är* (am/is/are) might very well

[1] the Göteborg Spoken Language Corpus (GSLC), (Allwood et al 2001)

resemble the verb *var*, but then *var* as an adverb may resemble *hur* (how).

In order to get around these problems, we used another approach, consisting of two steps – first we made the sibling measure, here called *sib*, symmetrical, so that

$$\forall w_1, w_2 : sib(w_1, w_2) = sib(w_2, w_1) \tag{2}$$

and then we collected clusters iteratively, i.e. after the first run the most similar words were replaced with the set of words representing their new found word cluster, and then we again looked for similarities. Thus if *gå* (go/goes) and *gick* (went) resembled each-other after the first run, then each occurrence of *gå* and *gick* were replaced by *gå-or-gick*, and then we looked for similarities again. This has several positive effects:

1. The clusters can be extended successively.
2. The risk of 'sibling chains' is much smaller.
3. Sub clusters fall out naturally, since the most similar words will be discovered first.

2.2 Adjustments in the sibling measure

Apart from the advantage with the symmetry property above in avoiding 'sibling chains', there is an intuitive appeal in having a similarity measure symmetric. One straightforward way to obtain this is to use the average of $sib(w_1,w_2)$ and $sib(w_2,w_1)$, see formula (3).

$$ggsib(w_1, w_2) = \frac{sib(w_1, w_2) + sib(w_2, w_1)}{2} \tag{3}$$

We also noticed that the sibling measure would become very low if one of the sums in the formula was low. Two words with an identical left context distribution and totally different right contexts will get a very small sibling value. We did not like this, so we changed the multiplication between the sums to addition and also divided the measure by 2 to keep the range [0,1]. With these changes, the resulting definition of the adjusted measure, called *ggsib*, will be:

$$sib(w_1, w_2) = \frac{1}{2f_{w_1}}\left(\sum_{w \in \overleftarrow{w_1} \cap \overleftarrow{w_2}} f_{ww_1}\left(1 - \frac{\left|\frac{f_{ww_1}}{f_{w_1}} - \frac{f_{ww_2}}{f_{w_2}}\right|}{\frac{f_{ww_1}}{f_{w_1}} + \frac{f_{ww_2}}{f_{w_2}}}\right) + \sum_{w \in \overrightarrow{w_1} \cap \overrightarrow{w_2}} f_{w_1w}\left(1 - \frac{\left|\frac{f_{w_1w}}{f_{w_1}} - \frac{f_{w_2w}}{f_{w_2}}\right|}{\frac{f_{w_1w}}{f_{w_1}} + \frac{f_{w_2w}}{f_{w_2}}}\right)\right) \tag{4}$$

The formula looks more complicated than it is, so we will take a look at an example to give an intuitive idea of how it works. Consider w_1=*han* (he) and w_2=*hon* (she). Now, formula (3) will give:

$$ggsib(han, hon) = \frac{sib(han, hon) + sib(hon, han)}{2} \tag{5}$$

Formula (4) tells us to add over all the words that occur immediately to the left and right respectively, from the words *han* and *han*. For example, one word occurring to the right is *har* (has). The frequency for *han har* is 394 and for *hon har* 242, but *han* is more common than *hon* (5269 and 2155 occurrences each). This tells us that the share of all *han* with *har* to the right is 7.4% and for *hon* it is 11.2%.

These relative frequencies fit well into in formula (4). For w_1=han, w_2=hon and w=har, we have:

$$1 - \frac{\left|\frac{f_{w_1w}}{f_{w_1}} - \frac{f_{w_2w}}{f_{w_2}}\right|}{\frac{f_{w_1w}}{f_{w_1}} + \frac{f_{w_2w}}{f_{w_2}}} = 1 - \frac{\left|\frac{f_{han\_har}}{f_{han}} - \frac{f_{hon\_har}}{f_{hon}}\right|}{\frac{f_{han\_har}}{f_{han}} + \frac{f_{hon\_har}}{f_{hon}}} = 1 - \frac{|7.4\% - 11.2\%|}{7.4\% + 11.2\%} \approx 0.80 \tag{6}$$

For identical relative frequencies, this will be 1, so each sum will result in a number between 0 and f_{w1} and thus $\forall\ w_1,w_2 : 0 \leq sib(w_1,w_2) \leq 1$, with the value 1 if all relative frequencies are identical. For *hon* and *hon*, the differences in relative frequencies for the most common context words are shown in table 1.

Table 1: Relative frequencies for the most common context words for *hon* and *han*.

Bigram	Value (6)	Rel. Freq$_{\text{han-bigram}}$	Rel. Freq$_{\text{hon-bigram}}$
att w_1	89%	8.5%	10.7%
och w_1	98%	4.8%	4.6%
men w_1	99%	3.3%	3.3%
som w_1	81%	3.1%	2.1%
har w_1	92%	3.0%	2.5%
när w_1	97%	2.9%	2.7%
ja w_1	87%	2.7%	3.6%
är w_1	92%	2.4%	2.9%
så w_1	95%	2.4%	2.6%
om w_1	94%	2.4%	2.1%
det w_1	91%	2.2%	2.6%
sade w_1	98%	2.0%	1.9%
w_1 har	80%	7.5%	11.2%
w_1 är	80%	7.4%	11.2%
w_1 var	96%	4.6%	4.2%
w_1 hade	85%	3.7%	2.7%
w_1 inte	90%	2.7%	3.3%
w_1 eh	81%	1.9%	1.3%
...			

We can see that the relative frequencies for the most common words around *han* and *hon* are very similar, which will result in a high ggsib-value.

2.3 Iterative use of ggsib

What we did in the iterative version of our experiments was the following:

1. We ran ggsib between all pairs of words with a raw frequency above a threshold Thr_{freq}. This gives a lot of similarity numbers between words.
2. We took the pairs with a score above a rather high score threshold Thr_{score}. Now we have a list *l* of pairs of rather similar words.
3. For each pair in *l*, we replaced the most low-frequent word with the other one in the pair.
4. If *l* is empty the Thr_{score} was slightly decremented.
5. We ran from the beginning again with the modified corpus until Thr_{score} goes below a limit.

With this method words with not so similar contexts will get more similar contexts after a while, and this will result in clusters of words. Within the cluster one could identify a structure because some of the words are put together early in the process and some later on. For example some of the pronouns are clustered like this:

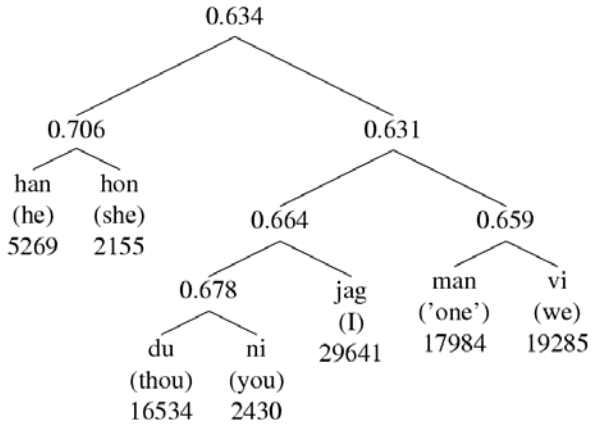

Figure 1: Cluster of some pronouns, showing the successive clustering.

The integers in the leaves are word frequencies and the numbers in the nodes are ggsib scores. We can see that the strongest similarity in this tree is between *han* and *hon*. These trees are always constructed bottom up, so *du* and *ni* form a small tree which is put together with *jag* and then the *man*-or-*vi* sub tree is added, and so on.

3. Implementation

The implementation of ggsib could be done quite easily, using for example Perl. For each word pair we just have to loop through all left- and right context words, and calculate the sums. But this will take a lot of time.

3.1 Time and space consumption

The problem arises when running this on a big corpus with a reasonable Thr_{freq}. In the GSLC there are 5828 word types with a frequency at of least 10, giving roughly 17 million word pairs to feed into the ggsib-function. For all these word pairs we will have to look up the frequency for all their context words. We do some slight hand manoeuvres to get rid of a substantial part of these lookups, but during the run we still look up roughly 218 million word-pairs in the lexicon. What we did was to speed up the lexicon lookup by using trees with arrays of letters in each node. This consumes a lot of memory, but the time is constant[2] to the corpus size. The complexity is still high, but in practice the example above takes less than an hour with a C-implementation[3], compared to four days with the Perl-script, using a hash table implementation of associative arrays. Maybe four days is not that bad either, but when running the iterative version our experiments would take almost a year to run.

[2]In fact it is linear to the average length of the words, but this will not change when the corpus grows.

[3]All time measurements are made on a SUN Enterprise 450 with 3 UltraSparc 300 MHz processors, running Solaris.

3.2 Time complexity

The theoretical worst-case time complexity is difficult to calculate, but if we assume that the Guiraud[4] value for a growing corpus is constant, it follows that $Types \propto \sqrt{Tokens}$ when the corpus grows. It also holds that $Types_{freq>Const} \propto Types$. The outer loops result in $Types^2$ laps, the loop in the sums will look up $Tokens/Types \propto Types$ words, and if the look-up takes $O(log\ Types)$, we will get a time complexity of $O(Types^3\ Log(Types))$ which is equivalent to $O(Tokens^{1.5}\ Log(Tokens))$.

3.3 Corpus details

We also had to think about utterance boundaries. After some tests we decided to insert a special utterance boundary word, *udl*, to avoid problems with utterance order and overlaps. Now, with the context window size of 1+1, each utterance is separated. This is the major reason to use this small context window size. We also had to decide how to handle sections of inaudible speech, and we simply decided to treat them as negligible in the calculation (we also removed the pauses).

4. Evaluation

The words are clustered the way they are only as a consequence to their distribution in our corpus. Figures 2 and 3 show examples of what the resulting clusters look like.

In some cases the resulting sets have a very clear semantic relation, as in the trees in figure 2:

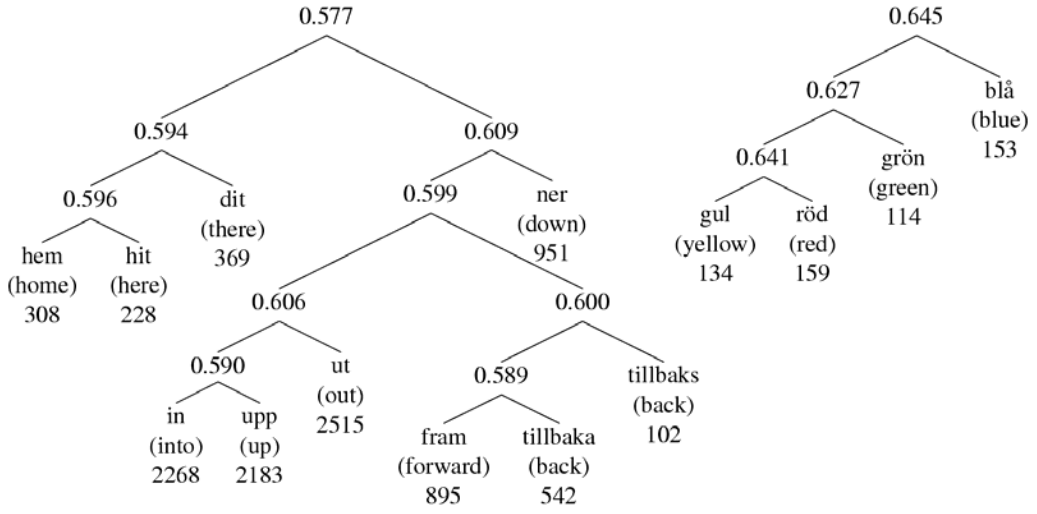

Figure 2: Two clusters with strong semantic coherence[5].

In some cases words with low frequencies are clustered in a semantically unexpected way, as in figure 3 (this will also happen if the score threshold is set low).

[4]Pierre Guiraud's widely used measure of vocabulary richness: $Guiraud = \frac{Types}{\sqrt{Tokens}}$ (Broeder et al 1993).

[5]The meanings of the words *hem*, *hit* and *dit* includes movement; *to home/here/there.*

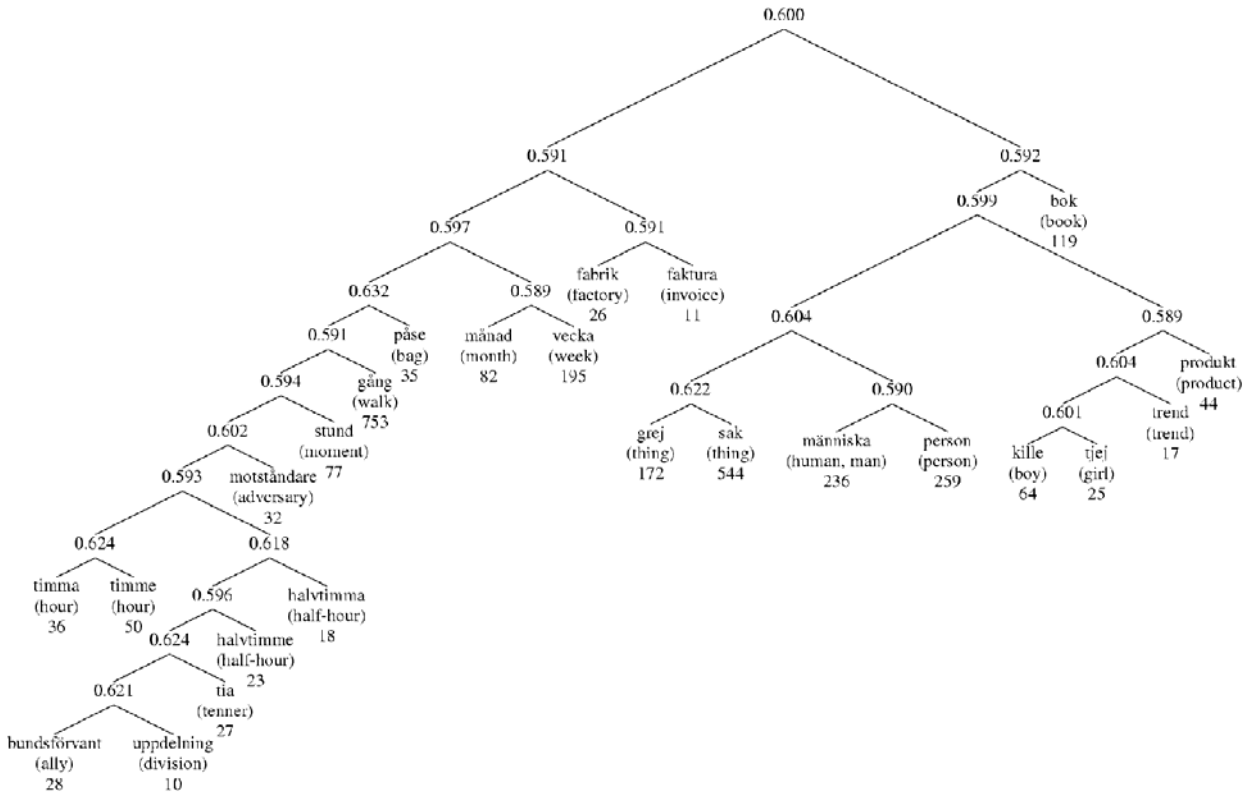

Figure 3: Cluster of rare words, with weaker semantic coherence.

Still, we see that all the words in this tree are countable nouns of common gender and indefinite form. Not even adjectives have managed to interfere.

Running the algorithm on a corpus in which pronunciation variants of the words have been transcribed, e.g. *talade* and *tala{de}* (spoke), where the curly brackets show reduced phonemes, these words are clustered first. Since this is what should be expected from a clustering method of our type, it is evidence of the validity of the method.

The tree with the strongest similarities is the one containing typical feedback words[6]. This can be explained by their typical occurrence as complete utterances, surrounded by the utterance delimiter (see 3.3 above).

We do not know of any acknowledged standard for evaluating word-clustering algorithms, and so it is difficult to see how our clustering would compare to other efforts.

The main problem with evaluating clustering is that it is not obvious what a correct result would look like. We have not defined an a priori taxonomy of words, but rather the result of the clustering *is* our taxonomy, and thus it is 100% correct[7]. But it is not entirely correct to say that we have no a priori taxonomy – our language intuition tells us that e.g. *he* and *she* should be clustered together earlier than *to* and *refrigerator*. That way it is possible to see some of the limitations of our algorithm, and the most obvious one is, as always, related to sparse data: for rare words, the clustering is not all that intuitive. E.g. *godafton* (good evening) and *ojdå* (oops), with 10 and 14 occurrences respectively, are joined in the very first phase. A plausible explanation to the clustering is that both words typically occur as one-word utterances, but our intuition tells us that they are not all that similar.

[6]Note that the algorithm works on categorematic as well as syncategorematic words (content and function words). The only requirement is that the frequencies are high enough.

[7]See section 5 for a discussion about what the clusters really are.

Another limitation of the algorithm is that homonyms, like *var* (was/were) and *var* (where) are not disambiguated, and that they cause certain 'confusion' in the clustering. In the case of *var*, it is more common as a verb, and so it is clustered with other verbs. In the next iteration, the adverbial use of the cluster containing *var* is even more rare, since the cluster includes other verbs, and thus the adverbial use of *var* disappears.

Despite these problems, the clustering is surprisingly good – even rare words are often clustered very well. E.g. the name *coronado*/13 is clustered with *silverado*/12 (the number after the slash indicates the number of occurrences); *reglera*/15 (regulate, control) is clustered with *tämja*/23 (tame, domesticate), and *fjol*/13 (last year) with *somras*/16 (last summer).

Comparing the results with the results presented in the article (Brown et al 1990), the clusters generated by our algorithm are at least as intuitive as the ones that Brown and his colleagues found. However, it would be interesting to do a more detailed comparison than what we could do from the Brown article only.

5. Intensional interpretation of the clustering

As mentioned in section 4 above, we have no underlying theory about which word clusters should be found, but the entire study would be rather pointless if we had no intensional description of the clusters that the algorithm finds. The traditional parts-of-speech – noun, verb, adjective, adverb, pronoun, conjunction, interjection and numeral – are distinguished (defined) using a mix of semantic, functional and morphologic criteria. E.g. *book* is a noun because it refers to a thing (semantic), *green* is an adjective and not an adverb because it is the attribute of a noun (functional), and *be* is a verb because it can be inflected as a verb. The criteria are not well defined, and not mutually exclusive, so that words like *intresserad* (interested) can be either a verb (perfect participle) or an adjective.

The clustering in our algorithm is based on the immediate neighbours of the words, a criterion which does not really fit into any of the three categories above. It does, however, approximate a functional classification, if we dare to assume that word order depends on grammatical function. Since the algorithm only looks at the immediate neighbours, one may object to the functional interpretation by saying that phrases may very well be longer than three words. But the immediate neighbours are themselves dependent on their neighbours, so that the clustering indirectly is controlled by a greater context.

Looking at the resulting clusters, the semantic similarity is striking:

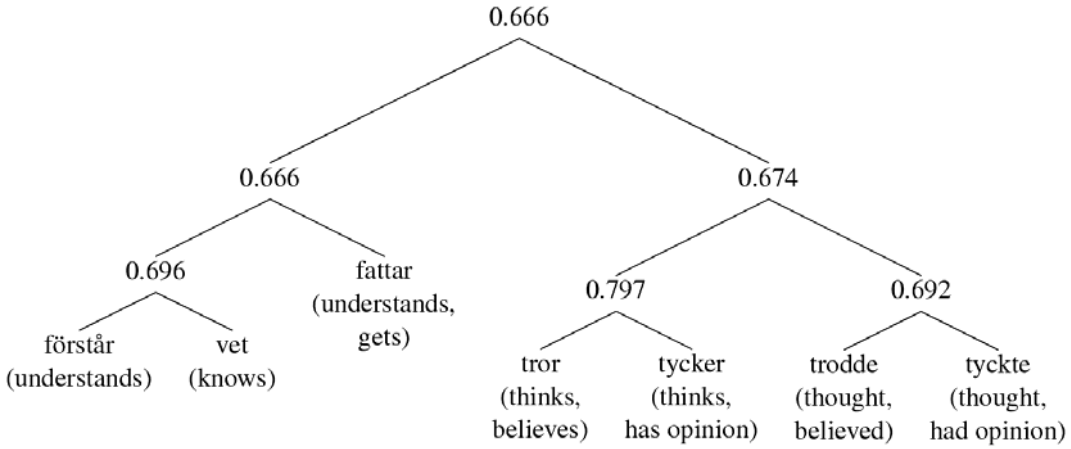

Figure 6: Cluster of some verbs for cognition.

This can be explained in at least two different ways. Firstly, there is the Wittgensteinian idea that the meaning of a word is to be found in its relations to other words in the language, which is a way of merging the semantic and functional criteria into one – the meaning of a word is a consequence of how it is combined and used with other words, or its grammatical function, if you like. Even if one does not agree with the strong version of this claim, but prefers a weaker version where the meaning and the function are

dependent, this is a useful explanation. Secondly, this algorithm is similar to the strategy often used to find synonyms, when the words in a defined window around a word are collected and treated as a bag of words. After this collection, the average bags of each word are compared to find similarities. The window in our algorithm is small (just one word on each side), but otherwise quite similar. The bag-of-words idea does not rely on Wittgenstein's assumption mentioned above, but rather on the idea that certain words will be more common when speaking of certain topics.

The morphological criteria traditionally used to cluster words do not show up in our algorithm. Different forms of verbs and nouns are clustered much later than unrelated words with the same inflectional form. There are of course exceptions; present and past tense (*presens* and *preteritum*) 'finds each-other' easily, as demonstrated by the tree above, and plural and singular forms of nouns are clustered early on as well. But when *varit* (been) is clustered with *är* (am/are/is), the *varit* cluster already contains 392 other words, including *fel* (wrong), *konstigt* (strange) and *solen* (the sun). The *är* cluster is by then still very much limited to verbs in the present and past tenses, but it contains 295 other words, including *viktigaste* (most important) and *anledningen* (the reason).

The great advantage of our clustering method is that it is to a high degree theory independent – the only assumption it relies on is that words with similar contexts can be exchanged.

6. Conclusions and further research

The clustering algorithm that we devised managed to find clusters that to a high degree correspond to our intuition, and this is true also for rare words. Since there is no way of finding out what would be the 'correct' clustering, a precise evaluation is not possible. Though the clustering is based on only the immediate neighbours of a word, the resulting clusters clearly show a high degree of semantic sensitivity.

We have identified two main areas where our clustering strategy can be improved. The first is that homonyms should be identified and disambiguated; it is not good as it is today, where the most dominant use covers the other uses.

The second area how to identify interesting 'sections' of the clustering – when should the clustering stop and return a set of clusters? Related to this problem is that of our rather arbitrary definitions of threshold values, and how much to subtract each time they are decremented. When studying the result files of the clustering, we discovered a pattern that might be used to determine when the clustering should stop. The diagram in figure 7 below is an effort to illustrate this.

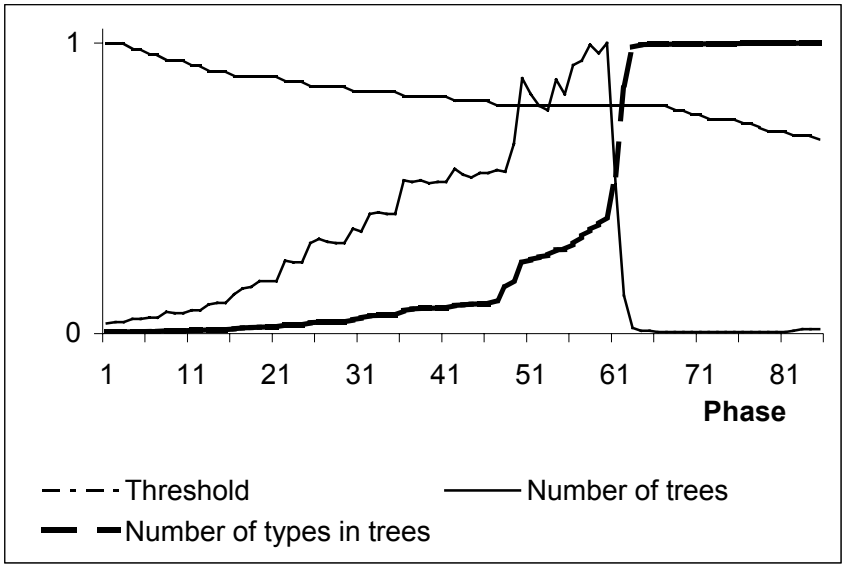

Figure 7: Number of trees and number of types in the trees during the clustering.

The values of the threshold, the number of clusters (trees), and the number of words in the clusters (number of types in trees) are plotted against the iteration (phase) number. The actual values of the three parameters have been normalised to the interval [0-1] for convenience[8]. The striking thing about it is the dramatic drop in the number of clusters around phase 61, and the corresponding jump in the number of words in trees. This is when some clusters have reached a size where they contain so many different words that their immediate contexts are 'any other word'. By then the algorithm collapses, and all the clusters are merged.

A natural end point for the clustering would then be to set the minimum Thr_{score} just above the value for the collapsing phase. However, a more precise analysis of this is necessary. There is no proof that this end criterion always works.

7. Bibliography

Allwood J 1996 Några perspektiv på talspråksforskning. In Thelander M (ed), *Samspel & variation, språkliga studier tillägnade Bengt Nordberg på 60-årsdagen*. Department of Nordic Languages, Uppsala Universitet.

Allwood J, Björnberg M, Grönqvist L, Ahlsén and Ottesjö C 2000 The Spoken Language Corpus at the Linguistics Department, Göteborg University. In *Forum Qualitative Social Research,* vol 1, no 3 - Dec 2000.

Allwood J (ed) 1999 *Talspråksfrekvenser – Ny och utvidgad upplaga. Frekvenser för ord och kollokationer i svenskt tal- och skriftspråk*. Gothenburg Papers in Theoretical Linguistics S 21, Department of Linguistics, Göteborg Universitet.

Allwood J, Grönqvist L, Ahlsén E and Gunnarsson M 2001 Annotations and Tools for an Activity Based Spoken Language Corpus. In *2nd SIGdial Workshop on Discourse and Dialogue Proceedings*, Aalborg, Denmark.

Broeder P, Extra G and Hout R V 1993 Richness and variety in the developing lexicon. In Perdue C (ed) *Adult language acquisition: cross-linguistic perspectives*, Vol. I: Field methods. Cambridge University Press, pp. 145-232.

Brown P F, Della Pietra V J, deSouza P V, Lai J C and Mercer R L 1992 Class-based n-gram models of natural language. In *Computational Linguistics* 18:467-479.

Gregersen F 1991 *The Copenhagen Study in Urban Sociolinguistics*, Vol. 1+2. Reitzel, Copenhagen.

Henrichsen P J 1997 *Talesprog med ansigtsløftning*. Instrumentalis.

Henrichsen P J 2003 *Siblings and Cousins - bilingual word-to-word mapping based on extremely narrow contexts*, IAAS, University of Copenhagen (Forthcoming).

Manning C D and Schütze H 2001 *Statistical foundations of natural language processing*. The MIT Press, Londin, England.

Nivre J 1999 *Modifierad standardortografi, version 6*. Technical report, Department of Linguistics, Göteborg University.

[8]In the diagram, the maximum value for the threshold is 0.75, for number of trees it is 199, and for number of types in trees 5389.