GUNNEL KÄLLGREN

Automatic Indexing and Generating of Content Graphs from Unrestricted Text*

1 Introduction

For quite some time, I have been exploring the surface signals of language, and trying to put them to as much use as possible, primarily in morphology-based part-of-speech assignment (Källgren 1984a,b,c, 1985) and pattern-based syntactic analysis (Källgren 1987). This kind of large-scale, probabilistic parsing on the basis of morphological and syntactic patterns has lately come to use in several projects. Some models that have been documented are the UCREL parser in England for the Brown and LOB corpora (Garside & Leech 1987), the VOLSUNGA parser in the USA for the Brown corpus (DeRose 1988), Ken Church's stochastic models (Church 1988), as well as other work in the US (Black 1988), but I am sure work along these lines is going on in several places.

The impetus behind the works just mentioned is mainly a need for analyzing large amounts of unrestricted text in a way that is not too resource-demanding, either on time or on computing power. As a secondary goal, I have seen the needs of large-scale information retrieval. Keeping my original surface-orientation, I have gone further from the analysis into parts-of-speech and constituents and started to look at the extraction and representation of some kind of 'content' from the surface of texts, without any kind of knowledge base support. This might seem quite impossible. Many of the other papers in this volume deal with how unavailable inferences are to the comprehension of text, and of course they are right. If the aim is to build a computerized system that will in any way simulate language understanding, it is necessary to have a large knowledge base and mechanisms for making inferences from it, but there are also applications where the human knowledge and inferencing capacities can be used instead. My approach in the experiment to be reported here has been to let each one do

*My sincere thanks to Boris Prochaska and Sten-Erik Bergner, formerly at Ericsson Telecom, who wrote the original version of the graph drawing program that I have used, and to Howard Gayle who set me in contact with them. Sune Magnberg has done a great job in transferring the program to a PC environment and has also written the part of the whole system that checks for collocational pairs. I also wish to thank Benny Brodda, Kari Fraurud, and Sune Magnberg for valuable comments on earlier drafts of this article.

Proceedings of NODALIDA 1989, pages 60-76

what he/she/it is good at; let the computer store, sort, and find facts, and let the human being do the inferencing.

This is actually the way it normally is in the field of information retrieval, where it is the human user that (interactively, at best) decides whether she/he has got the desired material. The problem is then to produce an optimal basis for that decision. I have so far been testing automatic indexing of texts, i. e. to find central concepts in texts automatically (Källgren 1984c). This is not all that difficult, but not all that effective either. In order to be covering, the index lists will easily grow too long; if shortened, important descriptors may disappear. This is an instance of what is known as the conflict between recall and precision. Still it is clear that simple word lists can be quite informative, though a bit boring.

I have also gone to the other extreme and tried to actually generate coherent abstracts automatically (Källgren 1988). This would certainly be desirable, and though it is far away, I do not think it is impossible.

What I will report on here is something in between. It is a way of showing central concepts and their interrelations in what I have called 'content graphs'. It is then up to the user to interpret the relations and make the inferences that are needed in order to get a picture of the content of the concerned text.

In principle, it may be wrong to talk about content graphs. The graphs picture 'what a text is about', rather than its content, but to call them 'aboutness graphs' is just too clumsy. What these graphs actually do is to give a hint about the content of a given text, not a full and true representation of it.

Given these limitations, the graphs still have their justification. The derivation of them from texts is an interesting task, for a set of reasons: The process can be fully automatized. It can be run on unrestricted text without manual preprocessing. The output can often be strikingly accurate. It also seems to have some interesting psycholinguistic implications.

2 Surface-Oriented Indexing and Information Retrieval

Of course, different kinds of surface-oriented methods have been used extensively through the years in research on automatic information retrieval. Salton & McGill (1983) give a broad overview of the field and also report interesting data on the notoriously difficult evaluation of information retrieval systems. Many of the systems described make use of adjacency and term frequency features in different combinations, and some systems take into account not only terms that are immediately adjacent but also terms that appear within a limited distance of each other (ibid. p. 33). Sophisticated methods for computing frequency and relative weight of terms occuring in documents are also described. Frequency of immediately adjacent terms is used as a means of finding complex terms (such as 'information retrieval'), but the authors do not report any work dealing with frequency of more loosely connected terms. The results of their experimentation is encouraging in that they show that well-constructed automatic

indexing systems may perform quite as well as manual indexing, and also that simple surface-based procedures can be as good as or better than more refined methods (ibid. p. 102).

The original inspiration for this work comes from Phillips (1985) which relates to some very early work within computational linguistics (e.g. Sinclair et al. 1970). Many of the ideas suggested at that time would deserve a renewed interest today, when computational power as well as linguistic knowledge has increased (the former considerably more than the latter, however). A good old idea that turns up every now and then is the concept of collocation. Collocations are words that appear together considerably more often than would be expected on purely statistical grounds. They can either be immediately adjacent or appear within a limited distance from each other. This distance, in terms of number of words, can be called a span. 'Collocation' and 'span' are basic concepts in my method for generating graphs.

To search for collocations can be a way of finding the idioms of a language, both those that are entirely fixed, like 'red tape', and those that contain slots, like 'pull someone's leg'. It can also be a way of finding relations between words. In the kind of content analysis that is carried out in the social sciences, cooccurrences between predesigned pairs or sets of words have sometimes been investigated. My treatment of the collocations is related to both uses; to the former in regarding all words in a text as liable to enter collocative relations, to the latter in assigning some kind of semantic load to the relations. This amounts to saying that the fact that two words cooccur suspiciously often carries some meaning in itself.

There is, however, one limitation on what words can form collocations in my system. To avoid uninteresting collocations, such as article plus noun etc., I only take content words into regard, not form words. The distinction between content words and form words is of course not totally clear (which linguistic distinctions are?), but clear enough to be operationalizable. There are some rare instances of homography, as when 'out' can be a noun in connection with baseball, and some adverbs can be felt to be 'content heavy'. Disregarding this, form words can be given as lists of words from the closed categories: pronouns, prepositions, adverbs, auxiliary verbs, articles, particles, and conjunctions. Removing such words from running text, or placing them on so-called 'stop lists', is a much used practice in automatic indexing, and it is estimated that about 250 common words cover 40-50 percent of an average English text (Salton & McGill p. 71). Lemmatization and rank ordering, as described in steps 2 and 3 in the algorithm below, are also well-established techniques.

3 The Algorithm

The method can now be presented in the form of an algorithm, which I will proceed to describe and examplify.

(1) THE ALGORITHM:

1. Eliminate all form words.
2. Lemmatize the remaining words, i. e. disregard differences at the end of words that either belong to the sets of derivational and inflectional endings or are admissible combinations of those.
3. Rank the lemmas in order of frequency.
4. Decide on a lowest frequency of lemmas and exclude the lemmas below that level. The level is dependent on the length of the text and the degree of recall wanted. The lemmas above the frequency threshold form the set of INDEX words.
5. Decide on a SPAN length. The length of the span is not dependent on text length or wanted recall, but might be language dependent.
6. Find all instances where two words from the index set appear within the same span. These are the COLLOCATIONAL PAIRS.
7. Find all pairs that are identical, disregarding order, as the pairs in themselves are unordered.
8. Rank the collocational pairs in order of frequency.
9. Decide on a lowest frequency of collocational pairs, based on the same principles as for the lemma frequency. Pick out all pairs above that frequency.
10. Construct ADJACENCY LISTS, i. e., for each lemma, list all other lemmas with which it forms a pair.
11. Use the adjacency lists as input to the GRAPH-drawing program.

Alternative version with graphs drawn by hand:

10′ Try to find optimal orderings of the pairs, look for central concepts that occur in many pairs.

11′ Draw the GRAPH.

4 Implementation

The system has been tested for Swedish, and the programs for removing form words and lemmatizing content words so far only exist in Swedish versions. They are a set of Lisp procedures running on PCs. For the purpose of demonstration, I will however use an English text where the lemmatizing has been done manually. The full English text is given in Appendix A and all examples below are taken from that text. Appendix B contains three shorter Swedish texts and Appendix C their respective graphs. These have been produced in a wholly automatical way as specified in the algorithm.

The removal of form words is, as stated above, simply done by removing all words on a pre-specified list from the text. Lemmatization is normally a far from

trivial process, but can in this connection be done in a simplified manner. The text, devoid of its form words, is treated as a word list and sorted alphabetically. Words that start in a similar way are compared as to their endings. If two words are identical all the way, they clearly belong together. If the parts where they differ belong to the pre-defined set of endings, they are also regarded as belonging together. This matching can be done in more or less sophisticated ways. Either the pairs of matched endings must signal the same part-of-speech and be morphologically connected, as when *berry* and *berries* form a lemma *berr* with matching endings *y* — *ies*. Otherwise, anything 'endinglike' will do, as when *favorite*, *favored* and *favorable* are matched. Actually, I think the latter alternative, lemmatizing across part-of-speech boundaries, should be preferred, as we are primarily looking for semantic relations, regardless of how they are expressed. In this way, the truncated stems (see below) that represent each lemma come to refer to a concept more than just a word. This kind of lemmatization has been called 'root lemmatization' and a linguistically sophisticated way of doing it is described in Fjeldvig-Golden (1984).

Semantically erroneous lemmatization can of course not be avoided, as when *late*, which in the text is used in the sense of *deceased*, is lemmatized with a temporal *lately*. This is however not such a big problem as one might suspect, as such infelicitious pairings rarely reach a frequency where they will influence the outcome of the entire process. To solve the problem, on the other hand, would demand a very large apparatus based on not only semantic but also pragmatic knowledge.

What is left when possible endings have been removed is a truncated stem, where the truncation process has sometimes been quite brutal. The truncated stems can now be sorted according to frequency and those below a certain level are removed. For the short sample text of two typed pages, a frequency level of two was settled. This is of course the minimal frequency. A frequency of one has no discriminatory effect whatsoever, as those lemmas can never occur in more than one pair. The lemmas with a frequency of two or more in the sample text are given in (2). They are called *index words* and are saved on a separate file to be matched against the full text in the next step of the process. For a longer text, a higher frequency level might have been preferred in order to limit the set of index words. This is a typical instance of balancing recall and precision to reach a result that is felt to be adequate.

(2) INDEX WORDS. (LEMMAS WITH FREQUENCY >= 2.)

acre	hair	orchard
berr	I	past
brown	includ	plant
captivat	North_Island	produc
chance	Jim_Macloughlin	seed
Chinese	kiwi	ship
commercial	kiwifruit	sold
countr	late	success
develop	lemon	tast
ear	like	Te_Puke
egg	market	thousand
favor	me	vine
five	millionaire	white
flesh	most	wild
fruit	new	world
green	New_Zealand	year

Next, a span length has to be settled. This does not, however, seem to be connected to recall and precision in the same way as the frequency limits. Rather, there seems to be an optimal span length. Increasing or decreasing the span length in relation to the optimal length will increase/decrease recall in the way that would be expected, while both increase and decrease of span length, interestingly enough, seem to reduce precision. An increase in span length will give more of accidental and thereby uninteresting collocations and also a higher relative frequency of such uninteresting collocations among all collocations above the critical threshold that is to be set in step 9 of the algorithm. At the same time, increased span length seems to give surprisingly few new 'hits', while the old hits run a risk of being outnumbered by the new accidental collocations. A decrease in span length will remove many wanted collocations, while the relative proportion of hits among the remaining collocations will not increase. Any variation of the span length thus seems to give a reduced proportion of semantically significant collocations. This is, however, only subjective impressions from small-scale tests with varying span length. Similar results have been reached by others (Sinclair et al. 1970, referred in Phillips 1985), and have led to establishing a span length of four orthographic words as optimal.

This is a point that would deserve a more thorough investigation. It probably has something to do with the normal size of common constructions: modifier and noun will almost always appear within less than four words distance, as will mostly subject—verb and verb—object, while e. g. more peripheral adverbials will not occur that close to the nexus part of the sentence.

In the sample application, the span length is settled to the optimal 4. The original text, including form words, is searched for occurences of the (truncated) stems of the index words. Whenever a word containing such a stem is found, a span of four words is scanned for more occurences of (stems of) index words. If

any are found, the resulting pairs are stored and the search goes on. In (3) a clause from the text is given with all index words capitalized.

(3) THOUSANDs OF ACREs ARE NEWLY PLANTED EACH YEAR IN A DOZEN OR MORE COUNTRIES, ...

Here, *thousand* collocates with *acre* and *new*, but not with *plant*. *Acre* collocates with *new* and *plant*, *new* with *plant* and *year*, and *plant* with *year*. *Countr* has no collocations in this instance. The internal order of the collocational pairs is of no importance, so the stems within each pair are stored in alphabetical order. The pairs are then sorted alphabetically and the frequency of each collocational pair is calculated.

The next step is again to decide on a lowest frequency, this time of collocational pairs. This decision governs which pairs, and consequently which lemmas, are to be regarded as representative of the content of the text. As this has such great impact on the output, it may well be that it should be possible to vary the frequency threshold for collocational pairs interactively, in order to facilitate closer inspection of interesting findings. A way of making expansions of the sets of lemmas and relations will be described below.

In (4), all collocational pairs with a frequency equal to or above 2 in the sample text are given in alphabetical order.

(4) COLLOCATIONAL PAIRS WITH FREQUENCIES.

berr — kiwifruit	2
commercial — kiwifruit	2
develop — kiwifruit	2
flesh — green	2
I — kiwifruit	3
I — New_Zealand	2

The idea is that this can give a more or less accurate picture of concepts and relations that are central to the text, at least in the sense that they show a high frequency. Mostly, this is sufficient to provide a hint about what the text is about. In some cases there may however be a need for enlarging the basis of the representation. This can be done by setting a lower minimal frequency level for lemmas or collocational pairs or both, but this means redoing parts of the processing. A better way can be to use a set of expansion operations as defined below.

Without changing the given frequency levels for lemmas and collocational pairs, we can derive the following sets of concepts and relations between concepts:

(5) EXPANSIONS

Primary concepts: the lemmas occuring in the pairs originally picked out by the algorithm.

First expansion: all collocations between primary concepts.

Second expansion: all other lemmas collocating with primary concepts. This gives the set of secondary concepts.

Third expansion: all collocations between secondary concepts.

Fourth expansion: all lemmas collocating with secondary concepts.

Etc.

The second and fourth (generally: all even) expansions are 'opening' expansions, as they bring in new concepts. The first and third (and all odd) expansions are 'closing', as they establish relations between existing concepts and make the corresponding graph more closely knit.

In (6) below, we see for each of the primary concepts the collocations it enters: a) with other primary concepts and with a frequency above the minimal level; b) with other primary concepts but with a frequency below the minimal level (first expansion); c) all collocations between primary concepts and other lemmas from the set of index words (second expansion). To construct all interrelations between all those items would in its turn give the third expansion.

From (6) we can also see that another characteristic of the collocations is their ability to delimit the interpretation of polysemous words. The pairs that a word can enter will often signal the specific meaning in which the word is used in a particular text. This is not so striking in this text as in some others, but looking at e g *green* we will see that we have to do with the green of fruit, not that of green paint, and *commercial* does not directly refer to e. g. banking, but to commercial aspects of growing fruit.

(6) COLLOCATIONAL PAIRS WITH FREQUENCIES: A) PRIMARY CONCEPTS, B) FIRST EXPANSION, C) SECOND EXPANSION

Primary concepts			Possible expansions		
kiwifruit:					
a)	3	kiwifruit I	b)	1	kiwifruit New_Zealand
	2	kiwifruit berr	c)	1	kiwifruit captivat
	2	kiwifruit commercial		1	kiwifruit chance
	2	kiwifruit develop		1	kiwifruit includ
				1	kiwifruit Jim_Macloughlin
				1	kiwifruit millionaire
				1	kiwifruit adjacency
				1	kiwifruit plant
				1	kiwifruit sold
				1	kiwifruit tast
				1	kiwifruit vine
				1	kiwifruit year
berr(y):					
a)	2	berr kiwifruit	c)	1	berr brown
				1	berr like
				1	berr tast
				1	berr wild
commercial:					
a)	2	commercial kiwifruit	b)	1	commercial New_Zealand
			c)	1	commercial orchard
develop:					
a)	2	develop kiwifruit	b)	1	develop New_Zealand
			c)	1	develop taste
				1	develop vine
flesh:					
a)	2	flesh green	b)	1	flesh I
				1	flesh kiwifruit
			c)	1	flesh tast
green:					
a)	2	green flesh	b)	1	green I
				1	green kiwifruit
			c)	1	green fruit
				1	green seed
I:					
a)	3	I kiwifruit	c)	1	I I
	2	I New_Zealand		1	I vine
New_Zealand:					
a)	2	New_Zealand I	c)	1	New_Zealand captivat
				1	New_Zealand lemon
				1	New_Zealand North_Island
				1	New_Zealand produc
				1	New_Zealand tast

Both (4) and (6) can, as said before, give hints about the content of texts if the lists are interpreted by a normally inventive human being. A graphic representation of the same facts seems, however, to be more striking and to facilitate

inference making. To proceed to this, a set of adjacency lists is constructed on the basis of (4). The adjacency lists form the input to the graph-drawing program, where each lemma will correspond to a node in the graph. In an adjacency list, each lemma, i. e. each node, is given a list of all its immediately adjacent nodes. This way, each collocational pair will be represented twice, corresponding to the two possible directions of the arc between the nodes. The graphs resulting from this system are however undirected. It would be possible to have weighted arcs in the graph, corresponding to the frequencies of collocational pairs, but this has not been implemented in the present system. The adjacency lists derived from (4) are shown in (7).

(7) ADJACENCY LISTS

kiwifruit(I, berr, commercial, develop)
berr(kiwifruit)
commercial(kiwifruit)
develop(kiwifruit)
flesh(green)
green(flesh)
I(kiwifruit, New_Zealand)
New_Zealand(I)

5 Graph-Drawing

The last step in the algorithm is the drawing of a graph. Automatic drawing of graphs by means of a computer is a demanding task, especially if the work, as in the present case, is to be done on a PC. We have, however, been able to find a satisfactory solution.

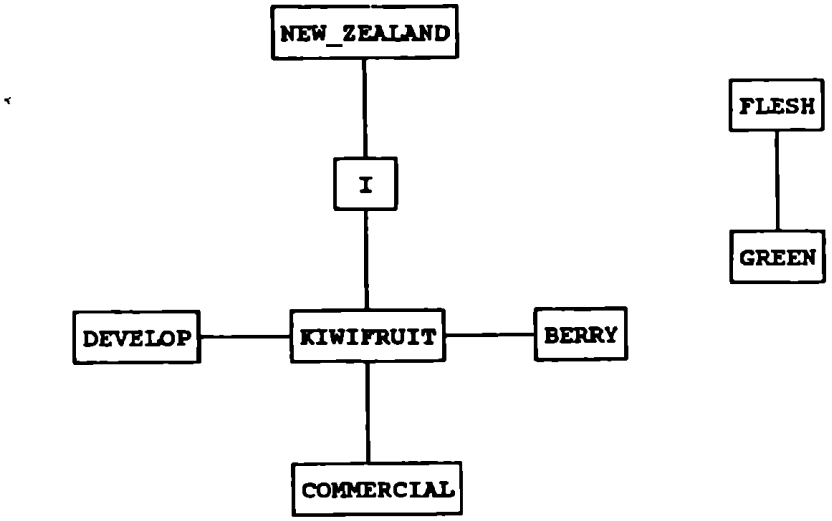
The program consists of two main parts. The first one finds the areas and subareas that together build up the graph. It tries to avoid crossing arcs, but if that is not possible, the program finds the best places to add 'pseudo-nodes', i. e. crossings. Its output is a 'road description' of the graph. The second part of the program performs the computationally heavy task of actually drawing the graph, laying it out nicely on the screen or in a file that can be stored or printed out on paper.

The first part of the program was originally written by Boris Prochaska as a part of his examination at the Royal Institute of Technology in Stockholm (Prochaska 1988), and the second part was written by Sten-Erik Bergner, who was Boris' supervisor during his examination job at Ericsson Telecom. Their version of the program is written in PSL-Lisp and had to be rewritten in GC-Lisp, a subset of Common Lisp, for use on PCs. This non-trivial job has been undertaken by Sune Magnberg, whose programming skills, earlier knowledge of graph theory, and general combination of inventiveness and patience, made the job of transporting the 'portable' Lisp possible.

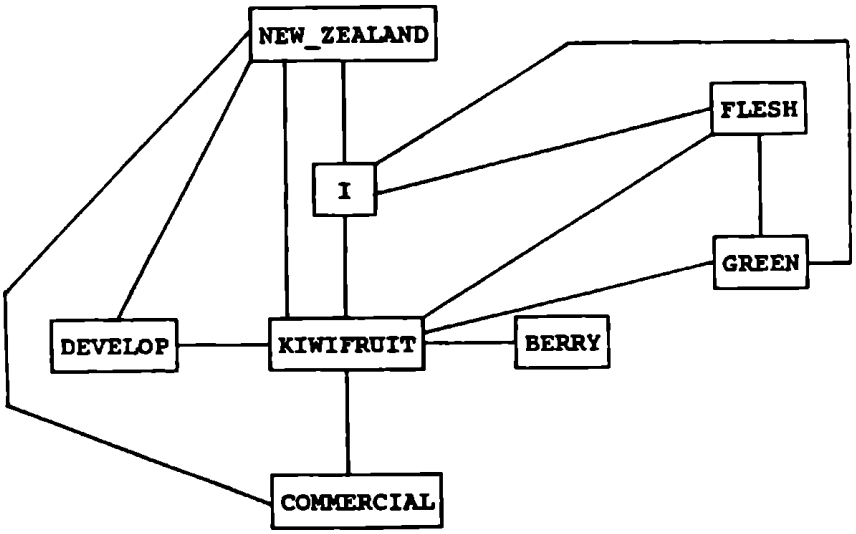
The result of all this is a graphic representation of the lemmas and relations that have a high frequency in a given text and, for that reason, can be assumed

to have strong connections to what the text is about. The graph that, by means of the described system, can be automatically derived from the text in Appendix A is shown in (8), while (9) is the first expansion of that graph (cf. (5)).

8



9



6 Concluding Remarks

These graphs support in principle the same inferences as did the lists of pairs, but in a neater way. The kind of relations that are signalled by the arcs varies considerably and is left to the human user to guess—at the risk of making mistakes. A very natural question to ask is whether all this apparatus gives anything more than would a simple list of high-frequency words. My impression is that it does. Below is a list of the 9 most frequent lemmatized content words in the text, all lemmas with a frequency of 4 or higher. The list should be compared to the words of the adjacency lists, (7), and to the full text in Appendix A.

(10) CONTENT WORDS FROM KIWI-TEXT, LEMMATIZED AND SORTED ACCORDING TO FREQUENCY

13	kiwifruit
10	New_Zealand
5	berr
5	ship
4	Chinese
4	vine
4	I
4	lemon
4	market

Kiwifruit, *New_Zealand*, *berry/ies*, and *I* are also represented among the eight lemmas picked out by the graph-constructing algorithm and the graph clearly shows their centrality. The graph also shows *commercial* and *development* as highly central, while the descriptions *green* and *flesh* are shown to be somewhat less central. The pure frequency statistics, however, has it that *ship/ping*, *Chinese*, and *lemon* are quite as important as *market* and *vine*. But the article (in Appendix A) is certainly not about the shipping of Chinese lemons, it is a subjectively colored boasting about the commercial success of kiwifruit and all that this has meant to New Zealand, interspersed with lyric bursts about the look and taste of the berry. There is no doubt that this is more clearly signalled by the graph than by the frequency list, although both representations need a good deal of human inference making to be added.

The results have not yet been independently evaluated, but the method has been applied to several Swedish texts. Three short Swedish texts are shown in Appendix B and their corresponding graphs in Appendix C. One very interesting finding is that the method seems to be utterly impossible on literary texts, but okey on others. Why this is so is something that has to be investigated more closely. It must also be investigated for which text types the method is best suited and under what circumstances it runs a risk of being seriously misleading.

Another step would be to try the method under realistic circumstances in connection with information retrieval. The idea is something like this: The user sits at a terminal and types in a search question, either in natural language, in which case it has to be parsed, or as a set of key words with or without Boolean operators. The key words are then matched against graphs that have

been previously derived from the texts in the data base to be searched. If the search question was in natural language, the presence of interrelations between key words can also be checked. A measure for when a graph is 'satisfactorily similar' to the information derived from the search question must be defined. Next, one selected graph at a time will be shown on the screen and the user can choose if she/he wants to have the full text. In doubtful cases it may be possible to get one or more of the expansions in order to get a broader basis for decisions. The search can also be carried out in such a way that graphs that are judged as relevant can be used for deriving new, conjoined graphs.

If these ideas can be developed to work well, the practical usefulness of the content graphs is clear, but among the most thrilling questions are why the method works when it works, and why it doesn't work when it doesn't. This is as yet far from clear.

References

Black, E., 1988. Grammar Development for Speech Recognition. Proc. from ELS Conference on Computational Linguistics, IBM Norway.

Church, K. W., 1988. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. ACL Second Conference on Applied Natural Language Processing, Austin, Texas.

DeRose, S. J., 1988. Grammatical Category Disambiguation by Statistical Optimization. *Computational Linguistics*, 14(1):31–39.

Fjeldvig, T. & Golden, A., 1984. Automatisk rotlemmatisering — et lingvistisk hjelpemiddel for tekstsöking. CompLex no. 9/84. Universitetsforlaget, Oslo.

Garside, R. & Leech, F., 1987. The UCREL probabilistic parsing system. In Garside, R. et al. *The Computational Analysis of English.* Longman, London.

Källgren, G., 1984a. HP-systemet som genväg vid syntaktisk märkning av texter. In *Svenskans beskrivning 14*, p. 39–45. Lunds universitet.

Källgren, G., 1984b. HP — A Heuristic Finite State Parser Based on Morphology. In Sågvall-Hein, Anna (ed.) *De nordiska datalingvistikdagarna 1983*, p. 155–162. Uppsala universitet.

Källgren, G. 1984c. Automatisk excerpering av substantiv ur löpande text. Ett möjligt hjälpmedel vid automatisk indexering? IRI-rapport 1984:1. Institutet för Rättsinformatik, Stockholms universitet.

Källgren, G. 1985. A Pattern Matching Parser. In Togeby, Ole (ed.) *Papers from the Eighth Scandinavian Conference of Linguistics.* Copenhagen University.

Källgren, G., 1987. What Good is Syntactic Information in the Lexicon of a Syntactic Parser? In *Nordiske Datalingvistikdage 1987*, Lambda, 7. Copenhagen, Handelshøjskolen.

Källgren, G. 1988. Automatic Abstracting of Content in Text, *Nordic Journal of Linguistics*, Vol. 11(1-2): 89–110.

Phillips, M., 1985. *Aspects of Text Structure.* North-Holland, Amsterdam.

Prochaska, B., 1988. Automatisk uppritning av grafer. Examination paper, Royal Institute of Technology, Stockholm.

Salton, G. & McGill, M. J., 1983. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York.

Sinclair, J. McH. et al., 1970. English Lexical Studies: Report to OSTI on Project C/LP/08, Dept of English, University of Birmingham.

A The Captivating Kiwifruit

Thirty years ago, growing up in New Zealand, I often sliced into a brown berry that looked like a duck's egg in a bristly hair skirt. Repulsive? Not really, for I knew a secret: The berry's odd appearance disguised an equally exotic interior, a sunburst of neat white streaks radiating from a creamcolored core, past tiny black seeds and into shimmering green flesh (above). Sweet-tart in taste, it seemed a succulent blend of strawberry, banana, melon, and pineapple flavors. Delicious! I loved the kiwifruit.

I still do, and today this peculiar product of a woody vine is captivating palates outside New Zealand at an extraordinary pace. In 1986 more than a billion kiwifruit, once called Chinese gooseberries, were tucked into trays and shipped to at least 30 nations. Thousands of acres are newly planted each year in a dozen or more countries, including the United States, France, Japan, and Italy, the leading producers after New Zealand.

This universal success has uniquely New Zealand roots. The kiwifruit's conversion to a commercial crop occurred in New Zealand, and its name—coined in the 1950s as a marketing tactic—conjures up both that likable country and its whimsical, flightless native bird, renowned for oversize eggs and hairlike brown feathers. Moreover, exports of the fuzzy, four-ounce berry are increasingly important to New Zealand's economy and the creator of more millionaires than anything else in my homeland's history.

The only fruit with such bright green flesh, the kiwifruit is one of just a handful of food plants domesticated within the past thousand years. Originating in the Yangtze Valley, it has long been a favorite of the Chinese, glorified in poetry as early as the eight century. Chinese peasants still gather the wild fruit for sale in rural markets.

The transformation of a small, hard, and wild Chinese berry into fleshier, tastier kiwifruit began about 1904, when a traveler returned from a China visit with seeds for Alexander Allison, a nurseryman on New Zealand's North Island. In the following three decades he and other gardeners developed superior kiwifruit vines through careful selection, pruning, and grafting. Most of these early fanciers were as much interested in the vine's showy white blossoms and attractive fan-shaped leaves as in its berries.

Kiwifruit farming got its commercial start in the 1930s, most successfully at Te Puke on the North Island's east coast. The late James MacLoughlin became the father of the modern kiwifruit—and ultimately a millionaire—by chance.

After he lost his job as a shipping clerk during the Great Depression, Jim's wife's aunt invited them to stay on her lemon orchard at Te Puke. "Later the bottom fell out of the lemon market," he told me, "but a neighbor sold the kiwifruit from a single plant for five pounds (then worth about $20 U.S.). To me that was a lot of money, so I risked putting in half an acre of them."

Luckily for MacLoughlin, the warm, wet climate and volcanic soil at Te Puke favored his vines. Neighbors soon launched their own commercial orchards, which further expanded during World War II when GIs stationed in New Zealand developed a taste for kiwifruit.

Then chance intervened again. In 1952 an English fruit importer ordered a shipment of New Zealand lemons. "To fill spare space in the ship, we included ten cases of

kiwifruit," Jim MacLoughlin explained. "A dock strike delayed the ship five weeks and the lemons arrived rotten, but the kiwifruit were in perfect shape." They sold well, and New Zealanders suddenly realized that they'd opened a world market.

B Swedish Texts

Text 1

Rudi och Renate hyr en liten stuga ovanför sjön, fast de har visst aldrig råd att betala den. Där finns ett rum och kök.

När Malin och jag kommer dit, sätter vi oss på golvet och jag tar av mig skorna också, jag vill vara som hon. Rudi spelar Mozart på en grammofon, som han lånat hem "på prov". Det är alldeles för dyrt att köpa egna grammofoner.

Solen skiner rakt in i köket. Rudi visar sina bilder, Malin röker pipa och ler så gott när hon ser tavlorna och Rudi pratar så mycket att jag slipper.

(Göran Tunström: Prästungen)

Text 2

Metropolen Oslo får en ny profil

Inte på 100 år har så många och omfattande byggprojekt påbörjats i Oslo. När de är klara kommer den norska huvudstaden att få en ny profil och nya möjligheter. Under tiden lider Osloborna.

Nordens högsta hotell, en kongresshall med plats för 10 000 åskådare och Europas längsta gågata under tak är några av de projekt som redan är i full gång.

Det har skett en snabb utveckling de senaste åren. Oslo blir alltmer en metropol. Vad gäller nattliv och restauranger kan staden konkurrera med både Stockholm och Köpenhamn. Den sista sammanräkningen visade 90 nattklubbar och kafeer som höll öppet mellan två och fyra på natten.

Aker Brygge med sin kombination av butiker, restauranger, teater och kontor i läckra omgivningar vid hamnen har blivit något som Osloborna stolt visar upp för tillresande. En bärande tanke har varit att öppna staden mot fjorden igen. Biltrafiken ska läggas så mycket som möjligt i tunnlar.

(DN 1987-12-05)

Text 31

Om batterier och batteribyte

Låt inte ett kvartsur som stannat bli liggande. Batteriet kan börja läcka och skada din klocka.

Vågar man då byta batteri själv?

Några få klockor har ett särskilt batterifack med lock, se bruksanvisningen. Då är det möjligt att själv byta batteri, men eftersom det kan vara svårt att få locket tättslutande igen efter bytet, är det klokt att ändå anlita fackmannen.

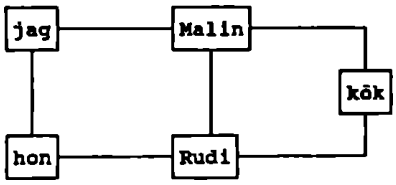
Det är också viktigt att du får rätt sorts batteri och inte ett som är avsett för fotoartiklar eller hörapparater. Då gäller inte garantin som de flesta tillverkare av urbatterier ger.

På de flesta klockor måste boetten öppnas vid batteribyte. Då fordras specialverktyg och stor försiktighet för att inte elektroniken ska ta skada. Och det är viktigt att boetten sluter ordentligt tätt efter batteribytet.

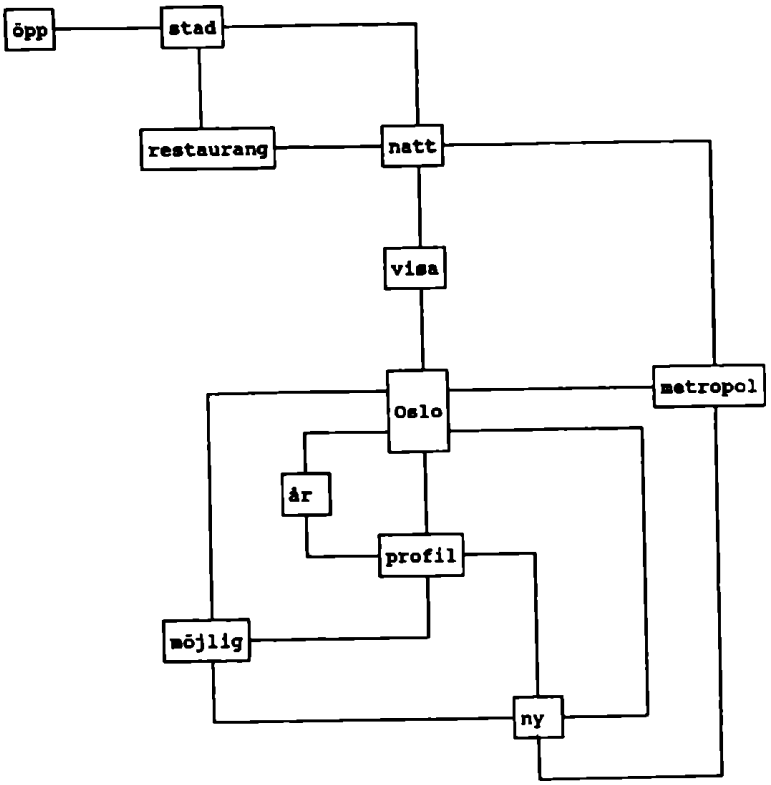
Det är ett arbete du ska överlåta till en fackman.

C Conceptual Graphs of the Swedish Texts

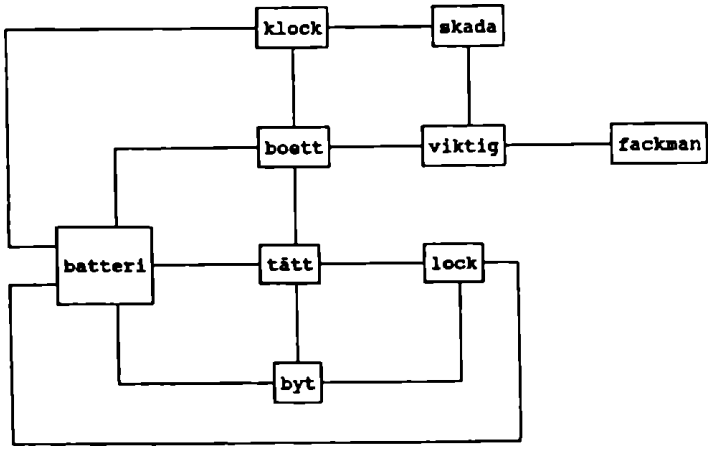
Text 1



Text2



Text3



Department of Linguistics
Stockholm University
S-106 91 Stockholm
Sweden