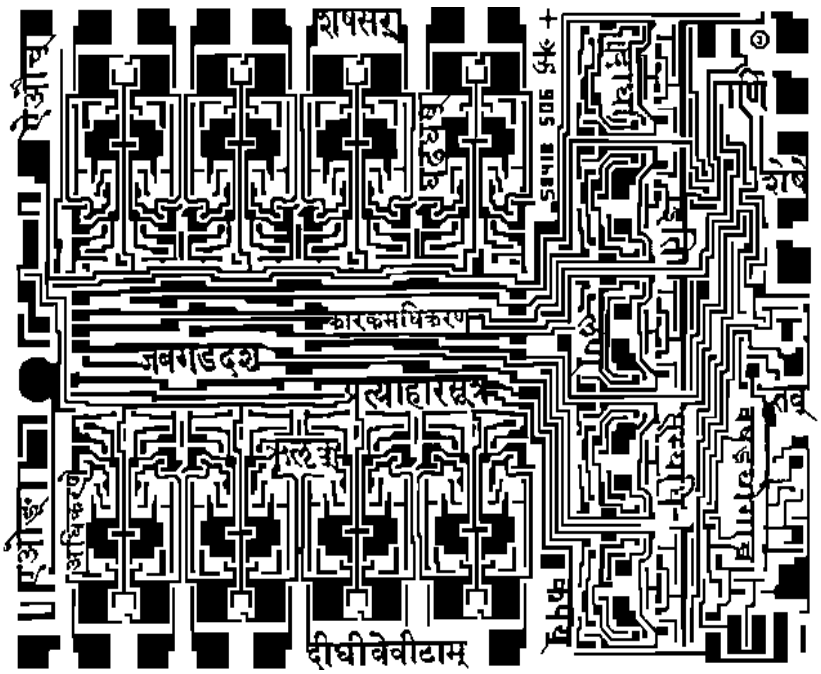
A Probabilistic Word Class Tagging Module Based On Surface Pattern Matching

Diploma Paper In Computational Linguistics
Stockholm University
1993



Robert Eklund

ABSTRACT

Title A Probabilistic Word Class Tagger Module Based On Surface Pattern Matching

Author Robert Eklund

CONTENTS

A problem with automatic tagging and lexical analysis is that it is never 100 % accurate. In order to arrive at better figures, one needs to study the character of what is left untagged by automatic taggers. In this paper untagged residue outputted by the automatic analyser SWETWOL (Karlsson 1992) at Helsinki is studied. SWETWOL assigns tags to words in Swedish texts mainly through dictionary lookup. The contents of the untagged residue files are described and discussed, and possible ways of solving different problems are proposed. One method of tagging residual output is proposed and implemented: the *left-stripping method*, through which untagged words are bereaved their left-most letters, searched in a dictionary, and if found, tagged according to the information found in the said dictionary. If the stripped word is not found in the dictionary, a match is searched in ending lexica containing statistical information about word classes associated with that particular word form (i.e., final letter cluster, be this a grammatical suffix or not), and the relative frequency of each word class. If a match is found, the word is given graduated tagging according to the statistical information in the ending lexicon. If a match is not found, the word is stripped of what is now its left-most letter and is recursively searched in a dictionary and ending lexica (in that order). The ending lexica employed in this paper are retrieved from a reversed version of *Nusvensk Frekvensordbok* (Allén 1970), and contain endings of between one and seven letters. The contents of the ending lexica are to a certain degree described and discussed. The programs working according to the principles described are run on files of untagged residual output. Appendices include, among other things, LISP source code, untagged and tagged files, the ending lexica containing one and two letter endings and excerpts from ending lexica containing three to seven letters.

Keywords Tagging, computational linguistics, word-class, probabilistic, morphology, swetwol, statistical, corpus linguistics, corpora, endings, suffixes, word class frequency, lexical analysis.

Language English.

Organisation Stockholm University, Department of Computational Linguistics,
Institute of Linguistics, S–106 91 Stockholm, Sweden.

Email robert@ling.su.se

Tutors Gunnel Källgren and Gunnar Eriksson.

Cover illustration *Sanskrintegrated Circuit*. Panini grammar tarsia on integrated circuit.

CONTENTS

ACKNOWLEDGEMENTS

Naturally, there has been a plethora of people assisting me in various ways and by sundry means. I have been given ideas, help, encouragement and advice as regards more or less all areas covered in and by this paper – linguistic approaches, programming, word processing, providing of articles and so forth.

Alas, would it were possible to include everyone who has contributed!

However, ere the preambulum commenceth, I needs must express my gratitude to at least the following, lest helping hands go unrewarded.

Johan Boye for interesting discussions on linguistics and some hints on lisp.

Mats Eeg-Olofsson, Staffan Hellberg and Ivan Rankin for kindly providing me with information on the existence of previously accomplished work in the field of morphological models.

Paul Sadka who refrained from turning my scribblings into decent English and kindly squeezed through as many archaisms as possible when sifting my ofttimes o'er-the-top pseudo-Elisabethan language, and also for putting up with my level-swapping idiosyncrasy and craving to circumvent readily understandable locutions wherever tangled alternatives were at hand.

Claes Thornberg for commuter-train lectures on computational science as extemporised as they were informal and ditto lisp advice, above all for having pointed out to me the existence and applicability of arrays! (Lazy evaluation is quite something, meseems!)

My course mates Janne ‘Beb’ Lindberg, Carin Svensson and Ljuba Veselinova for mental support, ‘guinea-pigging’ and encouragement!

Don Miller for heeding the call of a frantic author in need of last-minute language consultation.

Trumpet-blowing engineer Johan Stark and sitar-plucking indologist Mats Lindberg for help with the cover illustration.

Gunnar Eriksson for lisp advice, general guidance, warnings concerning the risk of positivistic world-domination tendencies in my language and for pointing out to me my obsession with trivial details.

Last, but definitely not least, Gunnel Källgren for constructive (and refreshingly stark!) criticism, encouragement and divine patience with my intermittent intellectual straying!

PREAMBULUM

As the man said, for every complex problem there's a simple solution, and it's wrong.
(Umberto Eco, Foucault's Pendulum)

Corpus linguistics has been an object of devotion in linguistics throughout time,[1] due to the inherent appeal corpora have to linguists of all flavours. Corpus linguistics, as Leech (1992) puts it 'combines easily with other branches of linguistics'. Recently corpus linguistics has become a fiancée of computational linguistics, and one might state – somewhat matter-of-factly – that one basic field in computational linguistics is the automatic tagging of corpora. Tagging, of various kinds, is a prerequisite for most parsing strategies, and as such a *sine qua non* for e.g. automatic translation. Tagged texts also provide a solid ground for more general linguistic studies of languages.

1989 saw the dawn of the *Stockholm-Umeå Corpus* project, referred to as the *SUC* (cf. Källgren 1990). SUC is conceived as a counterpart to the *Brown Corpus* (Francis and Kucera 1964) and the *Lancaster-Oslo/Bergen Corpus* (Johansson et al 1978). One of its goals is to collect a corpus of written Swedish containing at least a million words, eventually far more (see Källgren 1990). The corpus consists of a core part containing sentences with syntactic information and subsequently tagged words, and a reference part. The lexical and morphological tagging for the first million words is carried out by the SWETWOL lexical analyser at Helsinki (Karlsson 1992, Koskenniemi 1983a;b and Pitkänen 1992), and is complemented by manual inspection and tagging lest words or parts-of-speech be left with erroneous or no tags at all. The said manual inspection is also carried out to scrutinise SWETWOL'S lexical analyses, its ultimate objective being the ability to leave the tagging to computers alone with a good enough conscience for future expansion of the corpus.

After the machine analysis there remains, however, an untagged residue and the complete output can – somewhat harshly – be divided into the following subgroups:

1 – A group of unambiguously tagged words.

2 – A group of homographs given alternative tags.

3 – A residual group lacking tags.[2]

Whereas the second of the aforementioned groups is treated by Eriksson (1992), who describes an algorithm for probabilistic homograph disambiguation, the task undertaken in this paper is simply to tag the third, untagged, residual group.

Several tagging methods of various kinds and using different methods arrive at a success rate of circa 94–97 percent (Källgren 1992). A success rate of 95–99% is reported by Church (1988), 96–97% by Garside (1987) and 96% by DeRose (1988). A zetetic study of the remaining residual group is of course of utter interest if one wants to obtain better tagging figures. There are no obvious solutions to the problem. Eriksson (1992) has shown that augmenting the lexica does not solve the problem. As regards other possible solutions, a wide variety of strategies could probably be proposed. One method which could be tried (and indeed will be tried in this paper), is the use of an algorithm working exclusively on pattern matching, tagging words purely according to their surface appearance. Swedish is a language with a morphologically rich inflectional system, and it can thus be posited that a system working on a purely

[1] Although the actual term 'corpus linguistics' possibly was coined only in the 1980's by Aarts and Meijs (1984).

[2] There is a bulk of words which are never found in this group, preponderatingly those belonging to the closed words classes, since these normally are found in the lexicon.

morphological basis would provide a reasonably high rate of accuracy (Källgren 1992). Even in English, with a poorer inflectional system, one can obtain a high rate of accuracy from a morphological analysis (Eeg-Olofsson 1991).

Words in Swedish texts are – like words in any other language – ambiguous. Allén (1970) has shown that 65% of the words in a Swedish text are ambiguous. The corresponding figures for English are 45% (DeRose 1988) and 15% in Finnish. Since a given ending[1] may denote more than one word class, a graded output is called for, *scilicet* an output which provides information about all the possible word classes the morphological checker proposes, and each word class' respective probability. The latter is naturally very much dependent on the context in which a particular suffix appears and a decision needs to be made concerning the scope of the 'window' employed in the module. With what kind of horizon should one endow it? Since SWETWOL spits out output files of words on a word-for-word basis, thus ignoring (more or less) things like lexicalised phrases, particle verbs (ubiquitous in Swedish) and the like, by far the simplest solution is of course to equip the program with a one-word window. A conjectural supposition is that a higher rate of accuracy is to be expected if context is also considered, as attempts with purely heuristic parsers show (cf. Källgren 1991b;c, Brodda 1992). On the other hand, it can also be argued that there is palpable explanatory value in its own right in trying to find out how much information can be extracted from the words proper, neglecting their immediately adjacent text-mates. Moreover, as mentioned above, experiments with heuristic parsers have already been done; an implemented method of tag annotation of single words based on morphological surface structures alone, without any kind of lexicon is, to some extent, a breaking of fresh ground which could hopefully bring forth interesting results.[2]

The first thing that one has to do, however, is naturally to scrutinise with as great a punctilio as possible exactly what this aforementioned residual group actually contains. To this end a test run on text files covering 10 988 words was carried out, and an account thereof will be given in §§ 2.1–2.14. As was expected the material encountered is heterogeneous, and will thus be described in separate paragraphs. At the end of each paragraph a tentative solution will be discussed. However, not all problems encountered are to be solved here (or even attempted to be solved!), and more or less only the groups of untagged words which could possibly be taken care of by a morphology checker will be given tentative implemented solutions here, although a few other problem groups will also be accounted for.

---

1 All word-final letter combinations will henceforth, throughout this paper, be called *endings*, disregarding whether or not these be grammatical suffixes (or the like).

2 Descriptions of the relation suffixes/word classes for Swedish have been provided by e.g. Allén (1970), and models for English have been implemented, by e.g. Eeg-Olofsson (1991) and Rankin (1986), but so far only a few morphological models for Swedish have been implemented, e.g. Källgren's MORP (1992) and Eeg-Olofsson (1991).

1 PRELIMINARY NOTES ON THE TOOLBOX

After having sketched the incitements for this paper, a few comments on some of the decisions made might be provided as to the toolbox and 'wood' employed to carry out the work.

1.1 THE TAGSET USED

The success rate of any automatic tagger or analyser, *per se* and in comparison with other automatic taggers, is of course dependent on what tagset is being employed. The more general this is, i.e., the fewer the tags, the more 'accurate' the output will be, due to the lack of more subtle subcategories. Since the module described here is assumed to be included in a larger system, it is important that the tagset easily harmonises with the already existing tagset employed in this system. Since the *quasi*-ending statistics are obtained from *Nusvensk Frekvensordbok* (Allén 1970) – abbreviated NFO – I have here opted to adhere to the tagset employed in the NFO (cf. chapter 3 in this paper). The tags employed in this paper constitute a proper subset of the NFO tags and are shown in TABLE 1. It must be pointed out that NFO also employs tags for subcategories.

TABLE 1 – The tagset employed in the paper.

ABBREVIATION	WORD CLASS
ab	adverb
al	article
an	abbreviation
av	adjective
ie	infinitival marker
in	interjection
kn	conjunction
nl	numeral
nn	noun
pm	proper name (proprium)
pn	pronoun
pp	preposition
vb	verb
**	non-Swedish unit

Since it was found that not all words in the computer readable version of NFO were tagged, an additional tag was created to render the format consistent. Hence, the tag 'NT' is added, meaning 'Not Tagged (in NFO)'.

As to further specifications – such as gender, definitiveness and the like – these were eschewed, mainly due to the inconsistent format in which they appeared in the computer readable version of NFO.

1.2 A TEST RUN WITH SWETWOL

As mentioned in the preambulum, in order to investigate the untagged output, SWETWOL was run on a few texts, and the residual untagged output was collected in separate files, printed out and perused. (For referential information on the textfiles chosen, all of them belonging to the reference group of the SUC project, see Appendix F – Text Files.)

TABLE 2 describes the size of the untagged residual files and the percentages of tagged/untagged outputs. The term 'word' is liberally used here, since, as will be shown in the following, it does not always denote actual words, in the 'normal' sense, but quite often denotes parts of words, numbers, abbreviations et cetera.

TABLE 2 – Untagged residual files employed in SWETWOL test run.

FILE NAME	SIZE	TAGGED OUTPUT		UNTAGGED OUTPUT	
	WORDS	WORDS	PERCENTAGE	WORDS	PERCENTAGE
ADOP	52 634	52 443	99.6	191	0.4
ALPERNA	37 405	35 888	96.0	1 517	4.0
ANNA	123 657	122 961	99.5	696	0.5
DIREKTIV	14 940	14 836	99.4	104	0.6
DJUR	37 457	37 327	99.7	130	0.3
DN	38 001	37 145	97.8	856	2.2
DONAU	135 571	132 245	97.6	3 326	2.4
GALA	63 727	63 084	99.0	643	1.0
LOVSÅNG	58 251	57 990	99.6	261	0.4
MATTOR	34 960	33 719	96.5	1 241	3.5
OPERA	33 320	32 768	98.4	552	1.6
PARADIS	128 226	128 224	99.999	2	0.001
UNT	73 140	71 671	98.0	1 469	2.0
TOTAL	831 289	820 301	–	10 988	–

A description of the outcome will be given in the following chapter.

1.3 IMPLEMENTATION

Since the outcome of this paper is primarily intended to be linked to the SUC project, and since Eriksson's *Homograph Disambiguator* is implemented in COMMON LISP (Steele 1984, Tatar 1987), the source code language chosen for the module is also a Common LISP dialect. I have predominantly worked with PEARL LISP and ALLEGRO LISP on the Macintosh,[1] but also with LISP on Apollo work stations running under UNIX. By chosing LISP, the module will be easier to fit into the already existing system. It will also be easier to expand and/or change the module if future requirements show that this is called for.

[1] *Pearl LISP for the Macintosh*, Apple Computer, Inc. 1988-89; *Macintosh Allegro Common LISP 1.3 Reference*, Apple Computer, Inc.

2 A DESCRIPTION OF THE UNTAGGED OUTPUT

Already at first glance, one realises that the untagged material can be divided into a number of subgroups. One could of course give percentages of each specific group of untagged material, but this would be of little avail since over-representation of certain groups perforce is inherent in test runs of this kind. For instance, one of the untagged residual files used here treated the Alps, leading to an over-representation of Austrian and Swiss place nouns. Another untagged residual file was on carpets and mats, hence an over-representation of Arabic and Persian place nouns is discerned. It might be posited that a certain over-representation of any kind can hardly be avoided. As a matter of fact, the lexicon employed by SWETWOL covers Swedish and frequent foreign place nouns reasonably well, but naturally a line will have to drawn somewhere!

An example of an untagged outfile is given in Appendix B. The file ADOP has consistently been chosen for all full-length file examples (cf. Appendices), primarily due to its synoptical length, but also because it exhibits several of the problems discussed in the following. Naturally, not everything posing a problem to an automatic tagger shows in this output file, but a good enough general impression is surely provided, and might well serve its purpose as an introduction to the following discussion.

One may, however, pinpoint at least a few phenomena appearing, and a description of these will be given in §§ 2.1–2.14.

Before providing examples, a few comments on the format will be provided. In SWETWOL all letters are downcased. Majuscles are indicated with a prefixed superscript asterisk. SWETWOL outfiles have the words appearing in LISP bracket format, and special characters such a the Swedish *å*, *ä*, *ö*, French *ç* and German *ü* are represented by ASCII characters such as *}*, *{*, | et cetera. To facilitate reading, all examples have here been transliterated, and some of the special characters are thus replaced by their real-language counterparts.

Thus, an output word such as

```
("<*bourbon-*dampierre>")
```

... will here be written

```
Bourbon-Dampierre
```

Some of the phenomena to be described and discussed simply do not fit into one or other of the categories, i.e., it is not always possible to draw clear lines between the phenomena. Thus, a certain overlapping of the paragraphs will occur.

2.1 PROPER AND PLACE NOUNS

One of the first thing one encounters when one looks at what comes out untagged is proper nouns galore! This is not really surprising. Proper nouns do not to any greater extent exhibit any consistent morphological patterns.[1] Moreover, they abound, and it is hard to list them all in the lexicon. Liberman and Church (1992) mention that a list from the Donnelly marketing organisation 1987 contains 1.5 million proper nouns (covering 72 million American households). Since these have any number of origins, it is not feasible to cover them either with morphological rules or with a lexicon. Although the situation is somewhat less cumbersome in Swedish because of Sweden's more ethnographically homogeneous background, proper nouns

[1] Of course *some* consistent patterns can be found. Thus the suffix *-(s)son* in Swedish typically denotes a surname, as in *Eriksson*, *Svensson* et cetera.

of a great many origins do occur. Texts – e.g. translated novels – naturally include names a great many origins, and hence the problem is in one sense ‘international’.

Examples of output are:[1]

```
Casteel
Luke
Allavaara
Jokkmokkstrakten
Arabi-belutch
Azerbaijandistriktet
```

As can be seen, the above examples are of various kinds. Some are simple proper nouns, whereas others are ditto place nouns. Some are compounds formed by a place noun and a Swedish word, something which is very common in Swedish. Thus the compound word `Azerbaijandistriktet`, meaning ‘the Azerbaijan district’, is formed by `Azerbaijan` + `distriktet`. Whereas personal and place nouns appearing independently in a text could be solved by an increased lexicon (albeit far from easily concerning personal nouns, as already said), this cannot be done where this type of compound is concerned. Another thing to note is that the heuristic assumption that majuscle indicates proper nouns is confuted by the above listed words. It may also be argued that compounds like `Azerbaijandistriktet`, in fact constitute “real” nouns, and not primarily place nouns. This, however, more than anything, highlights the fact that the borders between the different problem areas are ‘fuzzy’.

A method proposed in this paper is to use what is here called the *left-stripping method*, in which an untagged word is stripped letter-by-letter from the left until the remaining word (the right-most residue) is found in the dictionary or in ending lexica containing endings with stastistical information regarding each endings’ word class set. (For detailed information on the ending lexica, cf. §§ 3 and 3.2.) The module employing this method is described in more detail in chapter 3. Since word classes (as well as gender) for compounds are typically attributed according to the last word in a compound, one may surmise that with this method it should be possible to annotate compound words correctly. Of course, this does not specify the type of material the first (singular or plural) parts of the compund is/are, but for annotation proper, this is of paltry importance.[2]

A problem will occur, however, if the infile contains corrupt words. Consider the two examples:

```
möbeloch
Bronsoijoch
```

Whereas the first word, `möbeloch` is made up of the two words `möbel` (‘piece of furniture’) and `och` (‘and’) with the space missing, the second is simply a place, `Bronsoijoch` (‘The Bronsoi Glacier’), a very common suffix in the Alps, `joch` meaning ‘glacier’. This is an example which happened to be highlighted here, since accidentally one of the untagged residual files was a traveller’s account of the Alps, as previously mentioned, but surely several similar problems will, and do, occur in most texts. A method that will actually see and decide when and where a given word actually *is* a word, and when it *is not*, is clearly beyond the scope of this paper's limited compass. However, I do not doubt that there should be a method for detecting this, since humans are able to make the said distinction! A possibility is obviously some kind of context viewing.

---

1 Unless other information is provided, all examples given will be obtained from the files used in the test run.

2 As for words found in the dictionary lookup, information is only given for the compound as a whole, and no information is given for the different parts of the compound words.

It is important to point out here that if a word like `bronsoijoch` is given the stripping procedure, it might well come out as some form of `och` (‘and’), which is a conjunction, an attribute that of course would make few glaciologists happy!

Also consider the following example:

`Fayet/St`

The above example is a mixture of what is here called a *slash compound* (using the slash to join two words together) and a split place noun. `St` (`Whatever`) is not seen as a whole, and thus a place like St. Tropez will be seen as two separate words, `St.` and `Tropez`. Since `St.` most often appears as a prefix to another noun, proper or place, information on this particular abbreviation could perhaps be included in the model.

The large amount of place nouns displayed in the untagged material here is of course due to the fact that one of the untagged residual files was a traveller’s guide, as already mentioned. The extreme solution to this problem is of course to expand the dictionary to include a detailed international atlas.

Something that could be considered here is majuscle heuristics in general. How much grammatical information is provided by upper case letters, initial or otherwise. Upper case letters appearing in texts might indicate a wide variety of different phenomena. Firstly, the first letter of each sentence in a typical text normally commences with a majuscle, regardless of what word class the word in question be. The untagged residual files treated in this paper, and consequently taken as input to the algorithm here employed, do not indicate whether or not the words were sentence-initial, and thus all words that actually *were* might lead the tagging algorithm up the garden path if any information as to word class is included in the ending lexica. Majuscles might also appear as e.g. Roman figures, in initials, titles and headings et cetera. Moreover, initials are used in different ways in different languages, and, for instance, the English habit of capitalising all words in titles (or all but prepositions, articles and conjunctions) is not employed in Swedish, which means that an English title in a Swedish text might ‘fool’ a tagging algorithm working according to Swedish standards. Contrary to Swedish, German capitalises all nouns and English all nationalities. I do feel that majuscles bring with them a problem of their own, and I have therefore chosen to let the algorithm exempt them, at least at this stage. For further discussion on majuscles, cf. e.g. Libermann & Church (1992), Eeg-Olofsson (1991:IV *et passim*), Källgren (1991b) and Sampson (1991).

2.2 ABBREVIATIONS

Several abbreviations are found in the untagged material. This is more surprising, since all these should, it is assumed, have been expanded/normalised in the pre-processing.

Examples are:

```
t.o.m.
cm2
km2
AC:s
enl
```

General problems here concern, among other things, *ad hoc* abbreviations, very frequent in texts. For instance: *I will henceforwards use the abbreviation ‘RM’ for the Reference Material earlier accounted for*. ‘RM’ would in this particular text constitute a noun phrase (i.e., NP), but unless the tagger actually *understands* the crucial sentence quoted above, annotation would fail since ‘RM’ is not an established abbreviation, and it would not be possible to tag either as a noun, or as an abbreviation. It might perhaps be hypothesized that abbreviations are often NP:s

if written with majuscles, something which could be accounted for in any proposed solution to the problem.

Another problem is that Swedish abbreviations can be given both with and without full stops. Thus, both `t.ex.`, `tex`, and `t ex` ('e.g.') can be encountered in a text. The obvious solution, however, is to include all established abbreviations in the lexicon. Non-established abbreviations are harder to deal with, and perhaps these will have to be tagged manually. A possible solution would of course be to hypothesise a word class given a syntactic parse. Nota bene! An abbreviation like 'RM' will indeed be tagged by the left-stripping algorithm proposed in this paper!

Yet another problem is all lexical abbreviations and acronyms, denoting organisations, unions, corporations, associations and the like. Acronyms like

```
WEA
MCA
RAF
EFTA
```

... and the like must be included as lexical entries in the dictionary. This, however, being just an ordinary lexicon coverage problem, is nonetheless great and further questions might be raised regarding possible confusions with majuscle initials in proper nouns.

2.3 COMPOUNDS

Compounds constitute a notorious problem in all automatic processing of Swedish. Compounds are ubiquitous in any arbitrary Swedish text, and a bulk of these compounds are of an ephemeral inclination, created on the spot for *ad hoc* purposes. These latter compounds are very productive and apart from having prosodics that differ from non-compounds,[1] normally have a semantic value which surpasses the sum of their constituents. Because they are legion, compounds constitute a very dire problem for any tagging module working on Swedish texts.

Occasionally it might be hard to decide where the compound border is located, since more than one alternative is available. Normally however, it is possible to establish where compounds have been joined simply by the appearance of allowed internal clusters.[2] In the current application, compounds cause problems: if none of the words of a compound are found in the dictionary, or if the TWOL-rules do not allow collocation. *Local disambiguation* (Eriksson 1992) handles over-generations of this kind. If there are several possible 'borders' in a compound, the alternative with the smallest number is generally the best (Karlsson 1992).

Another particular problem here is *tmesis*.[3] This, however, is not very common in Swedish, and consequently not considered here.

Here, several, different problems are encountered. These will be described one by one with illuminating examples.

---

1 Thus the difference between *central station* (adjective + noun) and *centralstation* (noun) can be heard easily, since they have completely different F_0 patterns.

2 For a detailed account thereof, see Brodda 1979. Funny examples of sandhi clusters are e.g. *prou**stskt skr**ivet* ('Proustly written'), *västku**stskt kl**imat* ('west-coastly climate') and (not sandhi, but genuine cluster) *skä**lmsktskr**attande* ('archly laughing').

3 That is, the interpolation of a word, or group of words, between the parts of a compound word.

One would imagine that the left stripping principle mentioned in § 2.1 (and chapter 3) should work without difficulty with compounds in which the last word occurs in the lexicon. Let us start with a word like:

`robinsonäventyr`

This is formed by the words `robinson` (‘Robinson’), and `äventyr` (‘adventure’). This word would be stripped of its leftmost letters, one by one, until the word `äventyr` could be found in the lexicon and tagged. As mentioned above, the word class of the preliminary words is of secondary importance, and need not be tagged.

A word such as

`folköverflyttningar`

... will give no problems. However, the following word will give rise to some ambiguities.

`heratimönstrade`

This could be interpreted either as `herat-i-mönstrade` (‘herat-patterned-in’) or `herati-mönstrade` (‘herati-patterned’). Note that the two different readings would possibly produce different tags if a finer tagging net were employed. ‘Local disambiguation’ (Eriksson 1992) should be able to cope with this problem.

Other compounds are e.g.:

`coverversion`

... formed by an English (nowadays also Swedish) word *cover* and a ‘Swedish’ word *version*. The word *version* will surely be found in a dictionary, but if not, it will be given a sufficiently correct tag by the left stripping algorithm, since its final letter combination is unambiguous enough to permit a satisfactory hypothesis for a noun.

`cooköarna`

... (‘The Cook Islands’), is formed by a foreign name, plus a Swedish word. Neither of these examples cause the tagger any inconvenience. It would not be a problem for the stripping method to take care of words like these. Examples like the latter will easily be solved using either a dictionary lookup, or by the left-stripping method. The word actually has two readings in Swedish, one being `cook-öarna` (‘The Cook Islands’), the other being `coo-köarna` (‘the coo queuers’). Even if we consider the fact that the word *coo* does not exist in Swedish, we might still see that the stripping method would provide both readings with the same tag.

Other similar examples are:

```
dirndlkjol
drangförfattare
leclerc-arméns
```

The last of these examples exhibits yet another feature of Swedish compounding: the compound `leclerc-arméns` (‘The Leclerc army’s’) loses its initial proper noun majuscle (as did `cooköarna`). Other examples which have been found are e.g.

`fnkonferensen`

... (‘The UN conference’), losing not only one majuscle, but two.

2.4 COMPLEX COMPOUNDS

A related problem is encountered in what is here called *complex compound*. By that I mean compound words created in ways diverging from the 'normal' compounding of two ordinary words. One example of this is when more than two words are compounded.

Instances of such compounds are:

```
Djursholms-Bromma-Lidingö-gängen
inte-jag
sånt-är-livet-filosofi
juan-complex
MBD-barn
BVC-mottagning
fot-i-fot
bra-känsla
XIII-sviten
karpatisk-balkansk-bysantiska
```

These clearly exhibit a *word-hyphen-word* pattern which could be formalized thus:

X-(Y-)*Z

... where the asterisk denotes an arbitrary number, possibly zero. To tag words of this appearance, one simply needs to check whether **Z** in the formalism is found in the lexicon, and tag accordingly. Compounds of the above type are typically written with hyphens, which is the case with 'normal' Swedish compounds, as can be noted in the previous paragraph. One counter-example has been found, however, since the word

```
jagvetintevad
```

... occurs in the output, where the 'normal' spelling perhaps would be *jag-vet-inte-vad* ('I-don't-know-what').

An implementation of a program which picks the last word of componds like the aforementioned has been made (cf. Appendix A). The incentive for including such a program would be mainly to save time, since in any case the left-stripping method arrives at the 'last' word eventually. The gain here would be but marginal, since when processing long untagged residual files, the time gained by skipping a few letters in but a few words cannot be of major importance, and thus this program is not included in the implementation of the algorithm.[1] Another reason for not including this routine is that a dictionary lookup could possibly succeed at an earlier stage than the final word (i.e., **Z**), and one would thus miss a dictionary tagging (even if this risk is perhaps minimal).

A similar problem concerns *slash compounds* (already touched upon in § 2.1) like the following:

```
Dannemora/Österby
Hornstein/Voristan
```

... where the slash ('/') separates two words according to the formalised pattern:

X/Y

---

[1] Garside and Leech (1982, p. 112), mention an algorithm which tags all parts of compounds, and in addition assigns an over-all tag to the compound word.

Words joined in this way typically belong to the same word class and therefore, either could be checked in the lexicon for correct annotation. A more certain method is, however, to check the final word. An implementation which choses the last word in slash compounds has also been implemented (cf. Appendix A). As was the case with hyphen compounds, this is once again primarily done to save time, since the last word would finally be reached. Hence, as above, the program is not included in the module. If one starts with the **Y** word, there is great likelihood that this will be found in SWETWOL, and one can therefore avoid further processing.

Other problems are encountered in words like

`göra-få`

This example – typical of psychological terminology – appears in an article on adoption and reads 'do-be allowed to'. We here encounter two verbs, which are to be read as a noun when compounded. Other examples of this phenomenon are:

`hungrig-mat-mätt`	reading	'hungry-food-full'
`du-och-jag-ensamma-i-världen`	reading	'you-and-I-alone-in-the-world'
`baby-på-nytt`	reading	'baby-anew'

Compounds like the ones above may be used typically in more than one way. They could be used as nouns, in for instance sentences like *This is a case of* '*hungry-food-full'*, or they can be used as adjectives in phrases like *He is a very* '*hungry-food-full' personality*. Hence it follows that the tagging is intricate, and that neither the stripping method, nor dictionary lookup would necessarily succeed. It is hard to see how a proper annotation of words of this type might be achieved without syntactic parsing. This also highlights another problem: the last word of a compound does not always indicate the word class, and one must ask whether or not it is possible to detect which of the words are more likely than others to be used in this freer way. (This problem is also akin to the meta-language problem, touched upon in § 2.12.)

The above examples were looked up in the original source, to see how they actually appear.

Inga samband mellan göra-få. ('No connection between do-be allowed to.') Here, obviously, the hyphen just replaces an 'and', to relate two verbs to each other.

Inga samband mellan hungrig-mat-mätt. ('No connection between hungrig-mat-mätt.') Once again, the hyphens seem to replace the conjunction 'and', and the words thus have their original meanings.

... ett annat du-och-jag-ensamma-i-världen. ('... another you-and-I-alone-in-the-world.') A clear noun, as is the last word of the compound.

... att bli barn-på-nytt, baby-på-nytt, nästan... ('... to become child-anew, baby-anew, almost...') Here the expression could be labelled a clear noun, being the complement of the verb 'become'. Note that the last word of the compund is *not* a noun!

For the module's proposed tagging of these, see Appendix G.

2.5 WORDS NOT IN THE LEXICON / SPECIFIC TERMINOLOGY

One problem concerns words that in all respects are 'normal', e.g. common, Swedish, established, morphologically clear et cetera, but for one reason or another are not included in the lexicon. Examples are for instance, in one sense or another, 'special' terms, belonging to specific domains. Specific terminology can be said to constitute a very normal part of the

language for initiated sub-groups of the populace, and words belonging to these groups are typically rare in all lexica but the biggest and most specialised. This problem could be avoided by using a bigger lexicon of course. One of the texts here used treated carpets and mats, and thus several ‘mat terms’ are encountered, of which a few are presented below.

```
gül
kantsyning
karderier
stadkant
stader
numdahs
våfflad
```

Interaction between the module and the user to augment the dictionary would be a good solution. The word is presented to the user, tagged, and then included in the lexicon. A manual check might of course be carried out by the group of linguists concerned, lest annotation not adhering to the group’s consensus be entered into the lexicon. This, however, is something that could easily be left to the discretion of a project’s responsible linguists, and need not be decided here. Another possible way to solve this problem is to have at hand several lexica, each one covering a certain field. These lexica could then be connected to the tagger when required, if it is known that a certain text treats a specific field.

The word `stadkant` (‘selvedge’) is an interesting example, since both *stad* (‘city’, ‘town’) and *kant* (‘border’, ‘edge’) exist in the dictionary, but require a joining *-s-* to form a compound.

2.6 DIACRITICA

Diacritica of all kinds present problems when they are seen through a one-word-window. Their function is not easily predicted, and in order to tag diacritical signs in a relevant way, context will probably have to be considered.

Examples (in SWETWOL format):

```
("<+>")
("<=>")
("<&>")
("<???>")
("<‹ ‹>")
("<+>")
```

How these are going to be tagged is a delicate problem. One could argue that the ampersand (‘&’) is a conjunction, but ‘+’ and ‘=’ do not really belong to typical word classes, and would more accurately be labelled operators. One solution would be to tag them as diacritica (e.g. “dia”). This would enable a search for diacritic in tagged files, which could be of interest.

In prose diacritics can be used in very creative and fanciful ways. The following excerpt from Herzog’s *Annapurna* presents a good example of what might be encountered in normal prose. How is one to tag ‘-?’ in a text context like the following?

> The villagers gathered round gesticulating:
> ‘Americans?’
> ‘No, French.’
> ‘-?’
> ‘Yes, French.’

As if this were conclusive proof, they nodded approval:
'Americans!'
'No, there are Americans, and there are Englishmen, but we are French.'
'Oh yes! But you are Americans all the same!'
I gave it up.[1]

Here it could be argued that '-?' constitutes an interjection, but how is the program to know? Supposedly, problems like these can only be solved by manual checking. Each project will probably have to decide how major a problem they consider examples such as the above to be, and how much time they are willing to spend on tagging these phrases.

2.7 NUMBERS

A great part of the untagged output is made up of numbers in various forms, exemplified below:

```
1 000--160
2 10,25x21,50
3 60--100x120--180
4 12-15-årsålder
5 8-uddig
6 1796-1820
7 -68
```

Whereas examples 4 and 5 may be resolved by the **X-(Y-)*Z** formalism (cf. § 2.4), examples 1–3 are more intricate. Numbers can denote various things, like for instance measurements, dates, weights et cetera. Example 6 denotes a period, and could thus be considered a noun, whereas example 7 (at least in Swedish) is used both as an adverb and an adjective (or even as a noun), depending on the context. A simple, Ockham's razor-like, solution, would of course be to tag them all as 'numerals'.

2.8 MISCELLANEOUS PROBLEMS

In order to give a more complete account of the untagged material envisaged in the test run, I will here briefly mention other types of untagged words. These do not generally constitute linguistic problems, but rather highlight some implementation problems to be solved.

Line-final hyphenation obviously makes the program miss words which are clearly to be found in the lexicon. Thus `("<fram->")` in what might be *framställning* ('account') is missed since the word form *fram-* is not listed in the lexicon, simply because the word appears thus:

fram-
ställning

... in the text. Other examples encountered in the test run are:

```
50-
bochara-
```

The solution here is simply to implement that should a line-final word end with a hyphen, it should be compounded with the first word on the following line, and then annotated according

[1] Herzog, Maurice, *Annapurna. Conquest of the First 8,000-metre Peak*. p. 56.

to the second word's annotation (the word class being decided by the last constituent in compounds). This should of course be dealt with during pre-processing.

Another, rather amusing, problem is posed by a word like

`aaaaahh!`

This word is of a recursive disposition which could be formalised thus:

$$\mathbf{a^{+}h^{+}!^{+}}$$

... where the plus sign denotes any number, equal to or greater than one, and not necessarily the same number in the three instances. A word like this could easily end with, say, five *h*:s, which would probably not be idiomatically represented in the ending lexicon. Finding a solution for words like these will probably prove rather problematic! A similar problem would, for instance, concern a word like *ooootroligt* ('iiiincredible'). Words like these (common in advertisements) fit the left stripping algorithm like a glove since they will generate a dictionary hit after three *o*:s have been stripped away!

2.9 ARCHAISMS

Archaisms often are used for various reasons. If not in treatises on Shakespeare, crammed with quotations, 'old' words might be used just 'for the fun of it', or to yield a certain atmosphere in the text. Incentives abound.

Archaisms found in the texts used here are:

`lif`	modern spelling:	*liv*	('life')
`af`	modern spelling:	*av*	('of')
`öfwer`	modern spelling:	*över*	('over')

Some of these old spellings are more common than others, and might be included in a big lexicon, whilst others may be very rare. One could of course implement spelling rules, so that forms like 'f', 'fw' and 'w' could all be exchanged for 'v' and then looked for in the lexicon. This method would more than likely be prone to over-generation, and needs to be carefully checked before employment.

A word like

`upptäcke`

... is hard to judge. Is it a simple spelling mistake, *upptäckte* ('discovered') or the more or less archaic usage of the subjunctive form *upptäcke* ('may discover'). A few subjunctives, in Swedish as in English, have been lexicalised, like for instance *Länge leve Kungen* ('Long live the King'), but most of them are not included in modern dictionaries (and in all circumstances, they do not have lexeme status). Rules for the formation of the subjunctive form in Swedish are, albeit not too complicated, at least more complicated than in English. Normally, the infinitive takes a final *-e* (if the infinitive already ends with a vowel, it is dropped). One could implement a program that checks all untagged *final-e* words for a potential subjunctive. This, however, is but a very marginal problem unless the text, as mentioned before, is a treatise on a 16th century author, or, even 'worse', an actual 16th century text! A clear example of a subjunctive form, however, is

`förbjude`

... meaning 'may forbid'.

2.10 FOREIGN WORDS

A large part of the untagged output is made up of foreign words, expressions and quotations et cetera. It is only natural that these have not been found during the lexical lookup phase during the SWETWOL run. Some English, French, German, Latin and Greek expressions have been lexicalized as Swedish expressions (for instance *rendez-vous*, *understatement*, *know-how*, *besserwisser*, *paëlla* et cetera), but the only blanket solution applicable here would of course be to enlarge the dictionary used, incorporating full English and French (et cetera) dictionaries. Interestingly enough, however, some of the suffixes used in at least the aforementioned languages, are sufficiently unambiguous to permit a graded tagging on the same basis as the one used for the Swedish untagged material. Thus, one can be fairly sure that words ending with for instance *-ium*, *-ukt* or *-tion* are of Latin origin, and that words ending with e.g. *–graf*, *-lit*, *-ark*, *-skop* or *-logi* are originally Greek. Moreover, several of these endings are typical of a certain word class. However, the ending tables which are the outcome of this paper will show percentages for those endings.

Examples of untagged foreign words encountered in the test run are:

```
the
au
revoir
again
Abbey
51st
54th
mkhatshwa
mnko
srbik
crkva
vrsac
```

Worthy of note here is that the lexicalised French expression `au revoir` is realised as two separate words. A tagging of these would give different annotation if considered as a whole. A solution attempted in the SUC project is to tag words of dubious origin as simply 'FOREIGN'. This, I feel, is hardly satisfactory, but considering that humans themselves cannot 'tag' foreign words[1] unless they are at least superficially acquainted with the language concerned, perhaps such a tagging will suffice for present purposes. It must be borne in mind, however, that the same problem also concerns Swedish lexicalised phrases.

The five last examples above clearly exhibit non-Swedish initial consonant clusters, and could thus instantly be recognised as foreign words, were it not for the fact that they may be used initially in compounds ending with a Swedish word. Hence, as regards tagging, word-*final* clusters are of greater interest, and these clusters could be compared with a Swedish 'cluster lexicon'. A description of such a lexicon is beyond the scope of this paper, however.

2.11 CORRUPT SPELLING

A problem which at first seems hard to solve is that of corrupt spelling in the untagged residual files. Computers do not have the same flexibility as human brains for seeing things from different angles when required. A few sentences from Liberman and Church are worth repeating *verbatim*:

[1] Unless tagging is based on a syntactic parse, hinting at a certain category.

> Humans can read text in all upper case or all lower case. They can even read text with upper and lower case reversed, or with S P A C E S between all characters. We+can#suddenly|decide-to replace%all^word*bound-aries!with some.random>characters?or evennone and humans can quickly adapt.[1]

Examples are as follows:

```
1 klövar.KATT
2 cr me
3 g~l
4 1950.Tekke-Bochera
5 garn.Erivan
6 avudnsjuk
7 chokartat
8 fascinderad
9 defenitivt
10 oosynlig
11 öpppna
```

These examples are of various kinds. Examples 2 and 3 include letters lacking representation. Whereas examples 1, 4 and 5 are simply cases of missed-out spaces between phrases (and thus should be dealt with during pre-processing), example 6 has two letters in swapped order. In lieu of `avudnsjuk`, *avundsjuk* ('jealous') should be found. `chokartat` ('shock-like') has one letter missing, the correct spelling being *chockartat*. `fascinderar` ('fascinates') has a superfluous letter; *fascinerad* is quite sufficient. `defenitivt` ('definitely') should have an *i* instead of the second *e*. Example 10 is rather uncanny, since `oosynlig` ('uninvisble') *could* be a deliberate neologism. This, however, is of little importance, since a morphology checker will provide a correct tag, misspelt or not. As regards examples 6–10, a module employing morphological information should be able to tag these words in a satisfactory way, since the endings themselves are not affected by the misspellings. If, however, the endings are affected, the problem takes a rather more gruesome turn! Of course one could tell the program to swap a few letters around until it 'finds something!', but this would also mean that, apart from (hopefully!) the right word, a great number of other plausible words would be found. That is, if the method would at all work! One can also argue that processing time for such a process would not be justifiable. A simpler solution is to say that words like `avudnsjuk` are simply not Swedish words, which of course is a correct judgement, albeit a somewhat 'easy option'. Examples 1, 4 and 5 should be quite easy to handle. One can formulate a rule saying that if an untagged word has a full stop within its scope, a space might be inserted, and then the two separate parts can be checked one by one.

Example 11 exhibits three identical adjacent consonants. Such dittographies are not allowed in Swedish, not even in compounds which would normally create such clusters.[2] This could easily be solved in the pre-processing phase.

2.12 NEOLOGISMS AND/OR EXPANDED USE OF WORDS

Human languages are typically very plastic. New words are continuously created, old words are given new interpretations, and are also used as members of other word classes. Thus, not even a word like *but* can be considered a sure-fire conjunction. In a phrase like *But me no buts!!!* it first occurs as a verb in its imperative form, and then as a noun in plural. (A Swedish, idiomatic, counterpart would perhaps be *Menna mig hit och menna mig dit!*, the story being a

---

1 Libermann and Church, p. 25.

2 Thus, a compound like *busstation* (bus station), which is formed by the words *buss* ('bus') and *station* (station) drops one *s* when forming the compound. If the word appears on two separate lines, or is hyphenated for any other reason, it is spelt with three *s*:s.

speaker annoyed with a listener who interrupts by saying *but* all the time!) Thus, we encounter words like the following in the untagged output:

```
1 nuen
2 måsten
3 CD-ns
4 r-ljud
5 somrigare
6 meetinget
```

Examples 1 and 2 are the words *nu* ('now') and *måste* ('must') in their noun/plural/indefinite forms (i.e., '*nows*' and '*musts*' respectively). Example 3 is the word *CD* in its definite genitive form, *ergo* an acronym with an added grammatical suffix. Example 4 should possibly be listed under the compound heading, meaning '*r-sound(s)*'. Example 5 is the adjective/comparative form of the noun *sommar* ('summer'), thus meaning '*more summery*'. Examples 1, 2 and 5 should quite easily be taken care of by the morphological module (working on untagged words), since *-igare* is a clear adjective/comparative form, and *-en* also a plausible noun/plural/definite. Example 6 is the English word '*meeting*' (Swedish: *möte*), given the Swedish neuter singular definite form.[1] This last word will also be given its correct tag by the module.

One must here point out that *all* words, irrespective of word class, might be used as nouns in a meta-linguistic way, for instance:

A 'green' would suit this phrase better!
Thou employest too many a 'lest' in thy prolegomenon, young esquire!

The border between what is 'expanded use' of word classes, and what is meta-language is of course fuzzy.

2.13 QUAINT OUTPUT

The test run also outputted some opaque material. Examples are given below (in SWETWOL format):

```
("< >")
("<>")
("<*>")
("<* >")
("<bengt-*d>")
("<*d*d-2:1-2-4-6>")
("<fra ois>")
```

Some of these obviously belong to a 'surplus crop' of formatting programs employed in the pre-processing, whereas the last is simply an example of a neglected letter (*c cedille*). However, since most of these examples have abstruse and arcane origins (albeit clearly not linguistic!), I have not taken them into consideration.

Other examples are for instance `de-cennier` ('de-cades'), a typical word processing formatting problem. 'Words' like these are to be expected, as long as people do not fully master word processing technique.

---

[1] Foreign words often obtain the gender of the corresponding Swedish word, for instance *practical joke*, which is neuter, obtained from the Swedish word for joke, *skämt*, which is also neuter (cf. Thorell, § 70).

```
("<1400>")
```

... is also hard to tag without its context (if not simply tagged ‘numeral’).

2.14 LEXICALISED PHRASES

Lexicalized phrases typically receive the wrong parses, especially if they allow other constituents to be included ‘inside’ them. Since, as mentioned before, the module here works with but a one-word window, these cannot be properly accounted for by the module, but can only be tagged on a word-for-word basis, tantamount to the SWETWOL process and Eriksson’s homograph disambiguator.

Examples are numerous, but since the output files serving as the basis for this paper include single words only, all lexical phrases will have to be ‘reconstructed’.

An expression like

```
au revoir
```

... serves well as an example.

2.15 GENERAL COMMENTS

Problems which were not highlighted in the test run output concern, among other things, non-transliterated quotations. How should a tagger deal with:

κακὸν ’ανάγκη, άλλ’ ούδεμία ’ανάγκη ζῆν μετὰ ’ανάγκμς[1]

Should a tagger be able to tag the following:

Вчера у нас были гости[2]

Phrases like the Greek one occur in academic texts while the Russian example is selected from a teaching text book. Although one should perhaps not expect a tagger to be multi-lingual, a tag for similar phenomena would perhaps be nice to have at hand.

As has been seen, there is a wide variety of input which may easily cause faulty annotation. Due to the fact that this paper works with but a one word window, problems like anacolutha and the like are not of primary interest. However, multi-word inputs render more important problem types other than those adumbrated above, and it must thus be borne in mind that the imbroglio of phenomena described in §§ 2.1–2.14 is not germane in its entirety to the following discussion.

The left-stripping module’s tagging of all the examples in §§ 2.1–2.14 is provided in Appendix G.

We will now proceed to describe the module proper.

---

1 *Evil is a necessity, but it is not a necessity to live with necessity* (Epicure).

2 *We had guests yesterday*.

3 MORPHOLOGICAL TAGGING ALGORITHMS

In this chapter the left-stripping algorithm proposed in this paper will be described in greater detail. The tagging problem has been approached by many a linguist in many a way, and, like a good many attempts have shown, perhaps there is no one model which will ever be able to solve the problem *in extenso*.

3.1 PREVIOUS RESEARCH

Morphological models of Swedish have been provided by Hammarberg (1966), Kiefer (1970), Linell (1972, 1976), Cedwall (1977), Hellberg (1978),[1] Brodda (1979), Blåberg (1984), Eeg-Olofsson (1991:III), Ejerhed and Bromley (1986) and others. These, however, predominantly treat either very specific areas of Swedish morphology with varying degrees of minutae, or are generative models for Swedish word formation. A somewhat different position is taken by the TWOL lexicon, which is bidirectional, and also used for tagging texts, thereby separating it from the other abovementioned models. However, although a lot of work has been done on the subject, little has been done in checking what comes through a model untagged.

Statistical methods in linguistics have been used since days of yore. At the end of the 19th century, Kaeding (1897) reported a corpus of 11 million words. In the '40s, Weaver (1949) proposed statistical methods for automatic translation. Probabilistic methods are used in speech technology, e.g. by Black (1988) and Jelinek (1986), and automatic translation, e.g. Brown *et al* (1990). Probabilistic parsing as such has been described by e.g. Sampson (1991) and Church (1987).

As for tagging, probabilistic/statistical methods in general have been used by e.g. Johansson and Jahr (1982), Marshall (1987), Garside and Leech (1982), Church (1987) and Garside (1987) in the tagging of the LOB Corpus. Eeg-Olofsson (1991:I;IV) describes a statistical model for word-class tagging, and DeRose (1988) treats grammatical disambiguation by means of statistical methods. Johansson and Jahr's project aimed at improving the suffix lists developed for the Brown Corpus by Greene and Rubin (1971); They basically worked by means of prediction of word classes in relation to grammatical suffixes, and to a certain extent also prefixes. Ejerhed (1988), Karlsson (1990), Koskenniemi (1990), Källgren (1988, 1991a;b), Magnberg (1991) and Eriksson (1992) employ probabilistic methods for lexical analysis.

3.2 A DESCRIPTION OF THE ALGORITHM

The algorithm employed in this paper (already briefly sketched in § 2.1) works by simple surface structure pattern matching, according to a method which could be called *left-most letter stripping*. The concept is to strip a word of its leftmost letter, look for the resulting word – *scilicet*, the previous word sans its first letter – in a dictionary (e.g. SWETWOL). If it is found, the word is tagged according to the dictionary, and the procedure is repeated with the next word. If it is not found, and the number of letters in the word is small enough to have a corresponding ending lexicon with that number of letters, the word is looked for in that ending lexicon. If it is found in an ending lexicon, it is tagged and the whole procedure is repeated with the next word. If it is not found, the word is stripped of what is now its leftmost letter, searched for in the dictionary et cetera. If no match is found even at the final (one) letter stage, the word is tagged thus:

("ENDING" ((NONE 0.0)))

[1] Implemented by Ivan Rankin (1986).

The rationale for this somewhat pleonastic design of the word class list is a desire to keep the format consistent. The flow chart in FIGURE 1 describes the module.

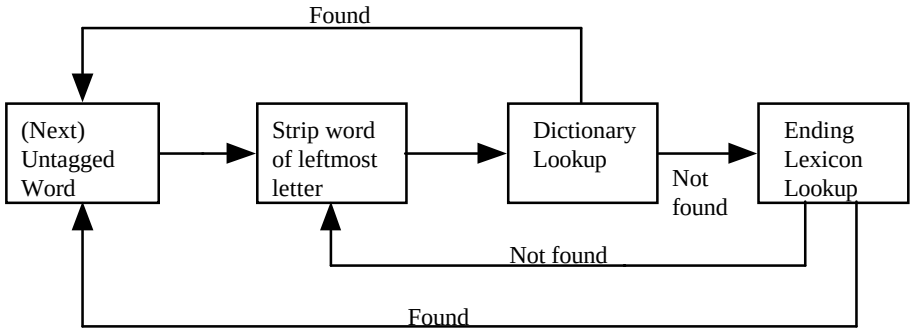


FIGURE 1 – Flow chart of the module.

As mentioned earlier, 'ending' here denotes the *n* final letters of a word, irrespective of whether these be grammatical suffixes, common combinations of any kind or unique word-final clusters.[1] Normally the dictionary lookup will succeed before the ending lexicon, since the length of a complete word normally perforce exceeds the length of its ending.[2] Thus, due to the

$$\text{length}^{\text{word}} > \text{length}^{\text{ending}}$$

...state of affairs, the module will make an exhaustive search of the dictionary employed, and only when the dictionary lookup fails will the ending lexicon see to it that the word obtains a 'probabilistic' tag. Grammatical suffixes by definition naturally constitute a proper subset of the endings in the sense that the word 'ending' has in this paper.

The module iterates over the untagged output list and strips the words recursively until a match is found in either the dictionary or the ending lexica. In the test run carried out here, no dictionary was employed, and the sub-routine intended to perform the dictionary lookup was foregone. The reason for this being the time it would have taken to iteratively search the TWOL lexicon in Helsinki. The important feature is the structure of the solution, and that the dictionary employed is given all possible chances before the ending lexica are employed.

3.3 OBTAINING THE ENDING LEXICA

A primary task in creating an ending list was naturally to find a relevant source to provide the material. In creating the ending lexica,[3] Greene and Rubin used a reverse-order listing of the *English Word Speculum* (Dolby and Resnikoff 1967). Johansson and Jahr also used the aforementioned source, as well as reverse-order listings of the Brown Corpus, the LOB Corpus itself and some other sources. A prerequisite, of course, was to obtain and implement information regarding relative probabilites of word class signalled by different word final letter combinations. To this end, the lists in NFO (Allén 1970) were of enormous benefit in obtaining a database of word final letter combinations, and statistics regarding the word classes associated therewith. NFO is a listing based on one million running words obtained from the

1 Benny Brodda (1979) has proposed the term *cadence* to describe word final clusters.

2 In some instances, however, it might be hard to tell the difference between a word and its ending. Thus, in quoting John Lennon's *Give Peace A Chance*: *...Ragism, Tagism, / This-ism, that-ism, ism, ism. ...* it might be hard to tell the difference between the word and the ending in *ism*.

3 A certain inconsistence as regards the numerus of 'ending lexicon/lexica' might be discerned in this paper. This is due to the fact that there is a choice between putting all the informatiion in one, big, lexicon, and in several lexica, one for each ending length.

material PRESS-65 and exists in computer-readable format. It might here be signalled that NFO is based exclusively on newspaper texts, and that other types of texts would perchance result in different ending lists. This is however a somewhat pleonastic statement, since results will always depend on what input material is being used. Hence, I do not consider it a major problem that the corpus is of so homogenous a disposition. Moreover, the primary issue here is to see to what extent this method might provide a higher success rate in lexical analysis, and only to a lesser degree to promote discussion on concerning the way in which certain endings relate to certain word classes.

A task here was of course to decide exactly how much was to be included in the model. If, e.g., suffixes were to be the basis of the model, which seems natural since most syntactical information is carried there, the question is what exactly counts as a suffix. As mentioned before, an initial decision was made to extract as much information as possible from single words. This means that if one weighs together co-occuring prefixes, which occur mainly in verbs, and some typical endings, one might obtain a higher rate of accuracy, since this procedure distinguishes some verbs from e.g. some nouns with the same endings, but without the typical verbal prefix.[1] Another detail here is that several 'typical' letter-combinations, in combination with a set of given affixes, hint at specific word classes in a very precise way. Thus, for instance, *-tion-* inside a word, followed by e.g. *-er*, *-erna* or *-ernas* produces almost 100-percent probability of a noun. If information regarding some of these 'pseudo-endings' were implemented, a faster lookup in the lexicon could be attained.

In comparison with dictionary lookup and grammatical ending identification, the left stripping algorithm has its pros and cons. How much do word final letter combinations – affixes or not – actually say about word classes? The concept of a recursive *word → stripped word → check* model would surely be more successful if word final letter combinations, other than grammatical affixes, were allowed, since this among other things, would mean, that common foreign words to some extent would be given accurate tags.

In order to obtain information on the endings that are connected to certain word classes, a series of LISP programs (see Appendix A) was run on NFO so as to obtain a listing sorted alphabetically according to endings. A typical entry in the computer readable version of NFO list looks thus:

```
"studieform" 1 "<>" "<studieform>" "nn" "-er" 1
```

Ending lexica were created with endings of 1–7 letters.[2] The word classes and their relative frequencies were calculated. Thus, the format was changed into the following:

("ENDING" ((WORD-CLASS$_1$ PERCENTAGE$_1$) (WORD-CLASS$_2$ PERCENTAGE$_2$) (WORD-CLASS$_n$ PERCENTAGE$_n$)))

Word class frequencies are given with four decimals, and the word classes appear in falling order according to frequency. Therefore, an authentic typical entry of the resulting list might look like this:

```
("ari" (("nn" 0.7802) ("ab" 0.1209) ("pm" 0.0934) ("**" 0.0055)))
```

One thing not considered was that in SWETWOL files all majuscles are downcased and prefixed with an asterisk (e.g. `("<*anna>")`).

---

1 One could of course find exceptions even to this rule. If one considers, for instance, the typical PREFIX+LEMMA+SUFFIX-verb "be-LEMMA-a", one can compare this with the noun *begonia*.

2 The number seven was chosen without any reason in particular.

The asterisks are however neglected in this paper since NFO does not employ asterisks in the same way as SWETWOL, and words like `("<*anna>")` above are therefore treated like any other words.

A first tentative test run provided some bizarre output. This undesired output proved to be due to some inconsistency in NFO, and the corrupt entries were simply lifted out of the reversed version. (For more detailed information, cf. Appendix D).

A final problem was the representation of certain numbers. Several endings with a number as the final 'letter' were not actual numbers but NFO representations of certain diacritical signs, for instance *accent accute* and *accent grave*. All these 'number-final' entries, where numbers have a diacritical role, were simply lifted out, in order to avoid an incorrect output. All entries lifted out of NFO are shown in Appendix D. Of course, this diacritical role of numbers appears in positions other than ultimate. For instance, the four-letter ending `re2s` is found in the four-letter ending lexicon, having the *accent grave* number 2 in the penultimate position. This is not very disturbing, however, since unless an untagged word actually exhibits this ending, it will simply not be matched to a given untagged word. Rather, it will be tagged according to the final *s*. Of course, this is also the case when diacritical number representations appear in antepenultimate or other positions.[1]

Another problem was that there were no instances of the figures *5* and *7* in NFO. These were simply added, using a copy of figure *9*'s entry. This manipulation can be defended by pointing out that all numbers presumedly have the same tag, "an" (abbreviation), and that there is no reason to assume that the behaviour of certain numbers is different from other numbers. For example, the number 8 entry is `b-18`, and the number 9 entry is `dc-9` (both likely to be aeroplanes). Given an extra order during the 50's, MacDonnell-Douglas would have made one more plane, and if so the DC-9 would now really be called the DC-10 (or whatever). I do feel that such considerations have little bearing on linguistics, and hence these 'manipulations' of NFO were carried out with a far from guilty conscience.

Another (albeit minor) problem was that UNIX automatically counted the number of instances of each (untagged) word in the SWETWOL untagged output files. These figures appeared to the left of the entries in the files. Prior to a module run, these figures were obliterated by a program written in the string handling language PCBETA (Malkior and Carlvik 1990), developed by professor Benny Brodda at the Department of Linguistics, Stockholm University.

3.4 SOME COMMENTS ON THE ENDING LEXICA

As mentioned above, I decided to run the programs up to a length of 7 letters. It is easy to check the number of entries the one-final-letter run yielded. It was 43 (cf. Appendix E). This means, theoretically, that with two final letters the number of entries would be 43×43 if all possible endings were to be found. With three final letters, it would be 43×43×43 et cetera. Of course, not all of these possible endings will be found; it is obvious that I have not been able to study all the endings in the ending lexica, and how particular endings relate to word classes. A study of the said lexica could surely merit a paper of its own! Hence, I will here only make a few sundry observations.

[1] An alternative solution would have been to preserve all these endings in the ending lexica for later manual matching. However, since SWETWOL files do not indicate accents in the same way, the 'take-it-away-solution' was preferred.

The numbers of entries in the ending lexica are shown in TABLE 5.

TABLE 5 : Numbers of entries in the ending lexica obtained from NFO.

	Number of letters of each ending lexicon						
	one	two	three	four	five	six	Seven
Number of entries in lexica	43	669	3 936	13 176	26 494	38 464	46 179

One thing which cannot be bypassed is of course the extent to which the number of word classes associated with an ending decreases with the number of letters in the ending, i.e., the longer the final letter cluster, the fewer word classes associated with that ending. Statistics indicating these relationships are shown in TABLE 6.

TABLE 6 : Number of word classes associated with numbers of letters in endings (percentages). Zero percent area is marked with bold line.

	Number of letters in ending						
Number of word classes	one letter	two letters	three letters	four letters	five letters	six letters	Seven Letters
One	30.2	39.5	52.9	71.4	85.4	92.1	94.9
Two	23.0	13.8	20.9	19.4	12.3	7.2	4.7
Three	0.0	12.1	12.9	6.2	1.8	0.6	0.3
Four	23.0	9.6	6.2	2.0	0.3	0.1	0.1
Five	4.7	7.3	3.3	0.7	0.1	0.0	0.0
Six	7.0	5.2	2.1	0.2	0.0	0.0	0.0
Seven	0.0	4.0	1.0	0.1	0.0	0.0	0.0
Eight	9.3	3.7	0.5	0.0	0.0	0.0	0.0
Nine	2.3	1.6	0.2	0.0	0.0	0.0	0.0
Ten	4.7	1.5	0.1	0.0	0.0	0.0	0.0
Eleven	11.6	0.7	0.0	0.0	0.0	0.0	0.0
Twelve	9.3	0.1	0.0	0.0	0.0	0.0	0.0
Thirteen	9.3	0.7	0.0	0.0	0.0	0.0	0.0
Fourteen	4.7	0.0	0.0	0.0	0.0	0.0	0.0
Fifteen	2.3	0.0	0.0	0.0	0.0	0.0	0.0

Whereas the curve is fairly even in the one-letter ending column, its inclination gets steeper and steeper with an increased number of letters in the endings. Also, zero representation within the higher number of word classes becomes more and more frequent in longer endings. This

implies that given, for instance, the 4 final letters of a word, it would be reasonable to exclude eight word classes not associated with that particular ending, since the table exhibits zero representation for that number of word classes. Moreover, given the five last letters of a word, we could with 85.4% certainty associate that word with one particular word class, and with 97.7% certainty associate that word with but two word classes. Which word classes these were would have to be looked up in the ending lexicon.

The steepness of the curves is shown in the following 3-D area chart.

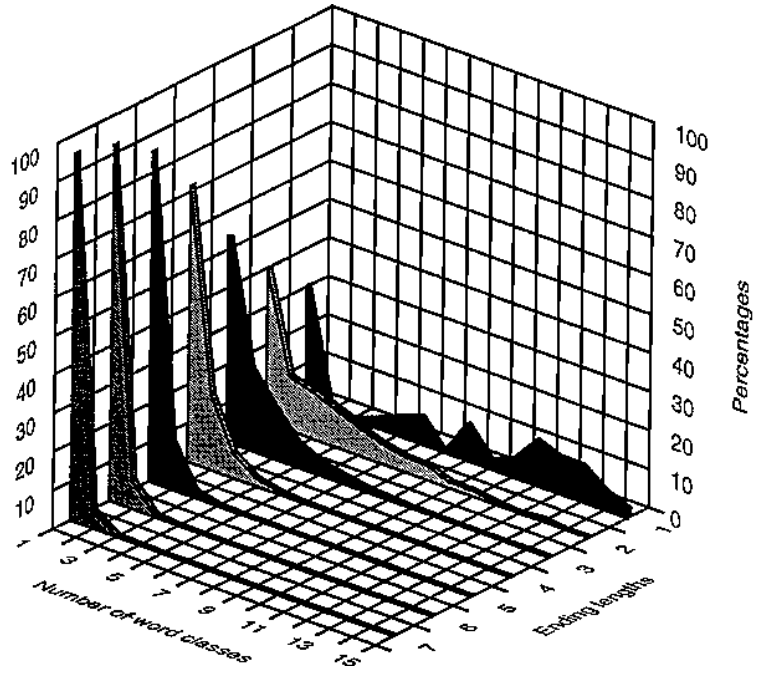


FIGURE 2 : 3-D area chart showing the steepness of the curves expressing the number of word classes associated with the number of letters in the ending.

It would be interesting to have a look at the contents of the lexica. Some endings are of course predictable in the sense that they constitute grammatical suffixes. As for these, in Swedish they can be added to one another. For example, a common plural marker for nouns is the suffix *-or* (first declension). The same declension's definite marker is *-na*, and genitive is most commonly marked with a final *-s* in Swedish. Thus we obtain the first declension paradigm with the pattern:

STEM+*or*+*na*+*s*

The corresponding paradigms for the second and third declensions are, respectively:

STEM+*ar*+*na*+*s*

STEM+*er*+*na*+*s*

An example word would be *dörr* ('door'), which consequently exhibits (if we ignore the singular definite form for the moment) the following pattern:

1 *dörr* “door”
2 *dörrar* “doors”
3 *dörrarna* “the doors”
4 *dörrarnas* “the doors’s”

One can see that whereas the stem is only vaguely marked for word class, *dörrar* is singly marked with a plural suffix, *dörrarna* is double marked with a plural suffix as well as with a definite suffix, and *dörrarnas* is triple marked. Thus, it could be posited that in the ending lexica, the noun (“nn”) list would obtain higher percentages for the triply marked line 4 than for the doubly marked line 3, which in turn would obtain higher figures than the singly marked line 2. Naturally, this pattern is a general one, as exhibited in the tables above, and is in a way tautological. Of course longer endings are less ambiguous markers since the shorter endings constitute a proper subset *vis-à-vis* the longer endings. It is nevertheless interesting to see the extent to which this is confirmed when looking at an *a priori* given pattern.

The following are excerpts from the ending lexica 2, 4 and 5 and show the respective word class lists of the endings discussed.

Two-letter ending lexicon:

```
("or" (("nn" 0.7478) ("av" 0.1217) ("vb" 0.0983) ("pm" 0.0221) ("**"
  0.01) ("NT" 0.0002)))

("ar" (("vb" 0.6909) ("nn" 0.2646) ("pm" 0.0161) ("ab" 0.0113) ("av"
  0.0108) ("pn" 0.0051) ("**" 0.0009) ("an" 0.0002) ("NT" 0.0)))

("er" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab"
  0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018)
  ("NT" 0.0003)))
```

Four-letter ending lexicon:

```
("orna" (("nn" 0.9692) ("av" 0.0235) ("pm" 0.0065) ("vb" 0.0008)))

("arna" (("nn" 0.9779) ("pm" 0.0181) ("vb" 0.0028) ("av" 0.001)
  ("NT" 0.0003)))

("erna" (("nn" 0.9493) ("av" 0.0356) ("pm" 0.0148) ("NT" 0.0001)
  ("**" 0.0001)))
```

Five-letter ending lexicon:

```
("ornas" (("nn" 1.0)))

("arnas" (("nn" 0.9841) ("pm" 0.0159)))

("ernas" (("nn" 0.9761) ("pm" 0.0239)))
```

The noun score for each of these endings are shown in TABLE 7.

TABLE 7 – Noun percentages (plural/definite/genitive) for noun declensions one, two and three.

	Paradigm according to the pattern *o*/*a*/*e* + suffix		
Declensions	*-r* (plural)	*-rna* (plural+definite)	*-rnas* (plural+definite+genitive)
First declension	74.8	96.9	100.0
Second declension	26.5	97.8	98.4
Third declension	41.5	94.9	97.6

As can can be seen, the percentages rise in the assumed way. However, the most dramatic rise is between the simple plural marker, and the plural + definite, especially as regards the second declension. This brings forth the observation that plural endings alone are obviously not 'sure fire' markers as to word classes, whereas plural suffixes + definite suffixes hover around 95% certainty.

An explanation as to why the *-ar* and *-er* suffixes obtain such low figures for nouns, is that these endings are the commonest verb present tense suffixes in Swedish, exhibiting the percentages 69% (*-ar*) and 30% (*-er*).

If we turn to single letters only, we are able to make some interesting observations.

First, the letter *u* has the following list in the one-letter ending lexicon:

```
("u" (("ab" 0.7944) ("nn" 0.0701) ("pm" 0.047) ("nl" 0.0257) ("**"
  0.0257) ("pn" 0.025) ("kn" 0.0058) ("an" 0.0054) ("NT" 0.0009) ("in"
  0.0002)))
```

What is striking here is the low figure for pronouns, 0.025. One of the commonest words in spoken Swedish is the second person singular pronoun *du* ('you'),[1] and consequently the low pronoun scoring here is somewhat surprising. The explanation is of course that the ending lists are obtained from newspaper material, and that the addressees of newspaper articles, even by the most pessimistic editors, are commonly seen as more than one single individual, which renders the pronoun *du* uncalled for in newspaper texts (unless used for special, personal effects). This is one example of the biased figures in the ending lexica explainable in terms of the character of the source(s). Beyond doubt more dubious figures could be detected if compared to ending lexica obtained from other text files, such as novels, letters (wherein *du* assumedly is more common a word!), text representation of spoken language and so forth.

What could also be commented on in the lexica of shorter endings is that certain endings obtain very high figures for certain word classes. An example is *z*:

```
("z" (("pm" 0.8687) ("nn" 0.0687) ("**" 0.0507) ("an" 0.0119)))
```

As can be seen, any word ending with a *z* in Swedish has an almost 87% proper name probability, as compared to 6.9% for nouns, 5% for foreign entity and 1.2% for abbreviations.

Källgren (1991b) lists some suffixes, being 'clear and unambiguous signals of word class', e.g. *-sion*, *-tion* and *-het* as noun markers. Looking these up in the ending lexica one finds that the

[1] Corresponding to the French pronoun *tu*, as distinct from its plural counterpart *vous* (*ni* in Swedish).

percentages corroborate Källgren's presumptions. Thus, the aforementioned suffixes' entries are provided:

```
("sion" (("nn" 0.971) ("**" 0.0237) ("pm" 0.0053)))

("tion" (("nn" 0.9647) ("**" 0.0281) ("pm" 0.0073)))

("het" (("nn" 0.9942) ("av" 0.0029) ("pm" 0.0017) ("ab" 0.0009)
 ("NT" 0.0003)))
```

As can be seen, the noun scores for these suffixes are high enough to justify the somewhat categorical label 'unambiguous noun marker', despite the fact that *-sion* reaches 2.4% for foreign entity (although *-sion* is quite likely a noun, even as a foreign entity!).

Källgren also mentions that *-tioner* and *-heter* invariably are nouns. Their respective entries are:

```
("heter" (("nn" 0.8487) ("vb" 0.1513)))

("tioner" (("nn" 0.9991) ("pm" 0.0009)))
```

Here we see something interesting. Whereas *tioner* satisfies Källgren's statement, *heter* fluctuates to some extent between a noun (85%) and a verb (15%). This is probably explained by the difference between *heter* and *-heter*, i.e. *heter* as an ending with preceding letters, and *heter* without preceding letters. The latter in this case a word in Swedish – the verb *heter* ('is called') in its present tense form. This may also point at one of the caveats of the module: that it does not differentiate between endings proper, and 'endings' coinciding or corresponding to words or word forms. One could of course argue that previous tagging is not likely to miss, for example, common verb forms, and that the module would therefore have to work on rather different premises. However, this is a characteristic of the module the employer–linguist will have to be aware of.

An untagged 'residue' file tagged entirely by the module is shown in Appendix C. To enable comparison between files, the file ADOP is once again provided, i.e. the file which was used to give an example of an untagged file (cf. Appendix B).

As mentioned earlier, a study on the ending lexica *per se* would surely disinter more than a few interesting finds.

4 DISCUSSION

As already argued, a discussion regarding success rates proper is perhaps a little out of place here. Since the module takes as its input but a minor residual group of untagged material, any result (save the completely inaccurate!) is perforce an improvement. Input files here constitute material which have failed lexical analysis by SWETWOL, and might therefore perhaps be considered 'problematic', in one way or another.

Moreover, since the module's output files here provide *graded* tagging, it is somewhat hard to discuss the results in terms of 'hits' or 'misses'. What could be discussed is how often the word class with the *highest* percentage is also the 'right', or desired, word class. This, however, is not all that simple to carry out, and would have to be checked manually by juxtaposing the original text with the tagged output file.

On the other hand, since output files provide graded tagging, interaction with Eriksson's local disambiguation algorithm (Eriksson 1992) might prove fecund. Future research might, or should, include attempts to link the two modules together.

The module is here used as an 'amendment device' to be used after a tagging process has already taken place, but it may of course be employed as a stand-alone module. It could be interesting to check how succesful it would be given such a status. As such, it would perhaps not arrive at very impressive figures, bearing in mind its inherent problem of biassing lists subject to the material used to put together the ending lexica et cetera. It would be very strange indeed if, *ceteris paribus*, tagging with the ending module alone would arrive at better figures than tagging obtained using a dictionary. To obtain a tagging tantamount to dictionary-based tagging as regards correctness, there would have to be a one-to-one correspondence between words, endings and their word classes. If we take word *w* and its ending w^{end}, then to arrive at the same success rate figures, w^{end} would always have to imply the same word class(es) as the word proper in the dictionary, which probably is not the case, since endings typically (at least the shorter endings) imply more than one word class. An evaluation of the module's performance when working on 'normal' input files will surely be of interest for further research.

Another thing that could be discussed is success rates regarding dictionary lookup hits when stripping words from their leftmost letters. However, the number of hits is naturally entirely dependent on the dictionary employed, and to what extent that particular dictionary includes, or does not include, atlases, foreign words, proper names et cetera.

It would be of great advantage if manual checking of this module's tagging could have an interactive connection with the dictionary being used, so as to enable the linguist–user to augment the dictionary with ease.

As to 'missed' foreign phrases and expressions, 'bizarre' neologisms and unaccounted-for remote villages in beyond-the-horizon countries, it must be borne in mind that most humans are not *a priori* familiar with all of the above, either, and that any formal system which is to be expected to handle all kinds of texts must include the set union of all human knowledge.

One thing that must be pointed out is that the ending lexicon was obtained from the processing of NFO only, and not from other types of texts. How large an error source this might be must here remain unrequited. The ending lexica provide empiric-probabilistic information, and in order to arrive at more accurate figures, one simply would have to base the ending lexica on more, and more varied, sources. Comparison between different corpora as to statistic word class information in connection with endings could be of interest here.

Eeg-Olofsson (1991:IV) raises the question whether it is at all desirable to assign categorical tags, thus foregoing the possibility of graduation in the description. I do not find it hard to

agree with his point of view. Examples where word classes are adequately vague (to me, at least) are easily found. For example, when a word traditionally seen as pertaining to a certain word class is used as a member of another category (examples given in paper).

Greene & Rubin (1971) use a special routine which matches certain prefixes with certain endings. This is a method which could not be included in the module provided, since our concept is based on leftmost-letter stripping, which by definition distorts and obliterates prefixes! If the word were not found prior to stripping, it would succeed only where a prefix occurred *inside* a compound word (i.e., later on in the process), and since it would need to be checked for at all levels, the increase in process time leads me to the conjecture that it is not worthwhile.

One interesting feature of ending-list based tagging is the method's inherent capabilitites regarding the tagging of *new words* (as pointed out by Greene & Rubin, *ibid*)! Since word formation obeys morphological rules, one may predict that neologisms and inflected loan words should be given rather accurate tags by the module.

Some consonant clusters are definitely not Swedish, and if such a cluster is encountered in a word, one could without hesitation annotate the said word 'FOREIGN' (given an exceptions list, including loan words like e.g. *mnemonisk*). This would mean that all words would be checked and compared with an exhaustive Swedish syllable lexicon. The process time for such an undertaking would perhaps not justify the increased success rate.

One of the contributions of this paper is the actual ending lexica *per se*. These have not been scrutinised in detail, but could presumedly provide interesting information if studied.

Fast process speed is always a desideratum (cf. Appendix A's Corollary). If text file lexica are employed, a useful strategy would perhaps be to store common endings in the beginning of the files. With hash-tables, the positioning of the entries does not matter as far as process-speed is concerned. This means that the trade-off between accuracy and process-speed which occurs when using text files, does not exist when using hash-tables, although the latter is more taxing for the RAM, which in turn is relieved by text file lexica. These considerations are happily left to the discretion of the linguist(s) concerned.

It would perhaps be interesting to see at *the level* at which words are tagged by the module, i.e., how many letters there remain of the word when a match is found. This may naturally be deduced from the percentages in the word class list, but an unambiguous label would perhaps be of benefit to the person doing the manual checking at a later stage.

Another point worth making is the module's limitations. Primo, it works on a *brute force* basis, rather than with linguistic *finesse*. The fact that it is not based on grammatical or morphological descriptions or models of Swedish precludes generation, whence it follows that the module is not bi-directional, a lack we will have to make do with, if we want to handle foreign entries. Secundo, as already pointed out, the ending lexica's information is perforce dependent upon the material on which they are based (in this case NFO). Tertio, tagging at lower levels, is graded. If an unambiguous tag is desired, the module must succeed at lengths greater than (in most cases) three letters.

Suffice it to say here that the module provides graded tags to untagged output files, and that the ending lexica *per se* could provide interesting material for further study to that motley group of scholars and scientists referred to as 'linguists'.

Explicit

REFERENCES

Aarts, Jan and Meijs, Willem. 1984. *Corpus Linguistics*. Rodopi, Amsterdam.

Allén, Sture. 1970. *Nusvensk frekvensordbok baserad på tidningstext (Frequency dictionary of present-day Swedish*), *1. Graphic Words, Homograph Components, 2. Lemmas, 3. Collocations, 4. Morphemes, Meanings*. Almqvist & Wiksell International, Stockholm.

Black, Ezra. 1988. *Grammar development for speech recognition*. In *Proceedings of ELS Conference on Computational Linguistics*. IBM, Norway.

Blåberg, Olli. 1984. *Svensk Böjningsmorfologi. En tvånivåbeskrivning*. Unpublished Master's Thesis, Department of General Linguistics, University of Helsinki.

Brodda, Benny. 1979. *Något om de svenska ordens fonotax och morfotax: iakttagelser med utgångspunkt från experiment med automatisk morfologisk analys*. PILUS 38, December 1979, Stockholm University.

Brodda, Benny. 1983. *An Experiment with Heuristic Parsing of Swedish*. Stockholm University, Dept. of Linguistics.

Brown, Peter et al. 1990. *A Statistical Approach to Machine Translation*. COMPUTATIONAL LINGUISTICS 16:2.

Cedwall, Mats 1977. *Semantisk analys av processbeskrivningar i naturligt språk*. Linköping Studies in Science and Technology, Dissertations No. 18.

Church, Kenneth Ward. 1988. A *Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text*, in *Proceedings of the Second Conference on Applied Natural Language Processing*. Association for Computational Linguistics, Austin, 1988, pp. 136-143.

DeRose, Steven. 1988. *Grammatical Category Disambiguation by Statistical Optimization*. COMPUTATIONAL LINGUISTICS, 14:1.

Dolby, James L. and Resnikoff, Howard L., 1967. *The English Word Speculum*. The Hague, Mouton.

Eco, Umberto. *Foucault's Pendulum*. Picador, Pan Books Ltd, London 1990.

Eeg-Olofsson, Mats. 1991. *Word-Class Tagging, Some computational tools*. Ph. D. Thesis including the following papers: I. *A probability model for computer-aided word-class determination*, II. *Software Systems for Computational Morhpology – An Overview*, III. *A morphological Prolog system for Swedish based on analogies*, IV. *Probabilistic word-class tagging of a corpus of spoken English*. Gothenburg University, institutionen för språkvetenskaplig databehandling.

Ejerhed, Eva and Bromley, Hank. 1986. *A Self-extending Lexicon: Description of a World Learning Program*. in Karlson 1986, pp. 59–70.

Ejerhed, Eva. 1988. *Finding Clauses in Unrestricted Text by Finitary and Stochastic Methods*. In *Proceedings of the Second Conference in Applied Natural Language Processing*. Austin, Texas.

Ejerhed, Eva; Källgren, Gunnel; Wennstedt, Ola and Åström, Magnus. 1992. *The Linguistic Annotation System of the Stockholm-Umeå Corpus Project. Description and Guidelines*. Version 4.31, May 22, 1992.

Eriksson, Gunnar. 1992. *Att två ord för att få en tolkning*. Department of Computational Linguistics, Institute of Linguistics, Stockholm University. (Forthcoming PILUS issue.)

Francis, W. Nelson and Kucera, Henry. 1964, 1971 and 1979. *Manual of information to accompany a standard corpus of present-day American English, for use with digital computers*. Department of Linguistics, Brown University, Providence, R.I.

Garside, Roger. 1987. *The CLAWS word-tagging system*. in Garside, Leech and Sampson (1987), chapter 3.

Garside, Roger and Leech, Geoffrey. 1982. *Grammatical Tagging of the LOB Corpus: General Survey*. in Stig Johansson 1982.

Garside, Roger, Leech, Geoffrey and Sampson, Geoffrey. 1987. *The Computational Analysis of English: a Corpus-Based Approach*. Longmans, London and New York.

Greene, Barbara B. and Rubin, Gerald, 1971. *Automatic Grammatical Tagging of English*. Providence, R. I., Department of English, Brown Universty.

Hammarberg, Björn. 1966. *Maskinell generering av böjningsformer och identifikation av ordklass*, pp. 59–70 in *Förhandlingar vid sammankomst för att dryfta frågor rörande svenskans beskrivning*. Sture Allén (ed.), Gothenburg University.

Hellberg, Staffan. 1972. *Computerised lemmatisation without the use of a dictionary: a case study*. In COMPUTERS AND THE HUMANITIES 6/72, pp. 209–212.

Hellberg, Staffan. 1978. *The Morphology of Present-Day Swedish*. DATA LINGUISTICA 13, Sture Allén (ed.), Department of Computational Linguistics, University of Göteborg, Almqvist & Wiksell International, Stockholm.

Herzog, Maurice. *Annapurna. Conquest of the First 8,000-metre Peak*. English translation by Nea Morin and Janet Adam Smith, Triad Paladin Grafton Books, London 1986.

Jelinek, Frederick. 1986. *Self-organized language modeling for speech recognition*. Manuscript, Continuous Speech Recognition Group, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, New York.

Johansson, Stig (ed.). 1982. *Computer Corpora in English Language Research*. Norwegian Computing Centre for the Humanities, Bergen.

Johansson, Stig, Leech, Geoffrey and Sampson, Geoffrey. 1978. *Manual of information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital computers*. Department of English, University of Oslo.

Johansson, Stig and Jahr, Mette-Cathrine. 1982. *Grammatical Tagging of the LOB Corpus: Predicting Word Class from Word Endings*. In Stig Johansson (ed.), *Computer Corpora in English Language Research*.

Kaeding, Friedrich Wilhelm. 1897. *Häufigkeitswörterbuch der Deutschen Sprache*. Steiglitz: Der Herausgeber.

Karlsson, Fred (ed). 1986. *Papers from the Fifth Scandinavian Conference on Computational Linguistics*. Helsinki, December 11–12, 1985. Publications No. 15, Department of General Linguistics, University of Helsinki.

Karlsson, Fred. 1990. *Constraint Grammar as a Framework for Parsing Running Text*. In Hans Karlgren (ed.), *Papers presented to the 13th International Conference on Computational Linguistics*, vol. 3, pp. 168–173, University of Helsinki, Department of General Linguistics, University of Helsinki.

Karlsson, Fred. 1992. *SWETWOL: A Comprehensive Morphological Analyser for Swedish*. pp. 1–45, NORDIC JOURNAL OF LINGUISTICS, Volume 15, Number 1, Scandinavian University Press.

Kiefer, Ferenc. 1970. *Swedish Morphology*. Skriptor, Stockholm.

Koskenniemi, Kimmo. 1983a. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. Department of General Linguistics, University of Helsinki, Publication No. 11.

Koskenniemi, Kimmo. 1983b. *Two-Level Morphology for Word-Form Recognition and Production*. Department of General Linguistics, University of Helsinki, Publication No. 13.

Källgren, Gunnel. 1990. *"The first million is hardest to get": Building a Large Tagged Corpus as Automatically as Possible*. In Karlsson (ed.), 1990.

Källgren, Gunnel. 1991a. *Storskaligt korpusarbete på dator. En presentation av SUC-korpusen*. In SVENSKANS BESKRIVNING 18, Lund University Press.

Källgren, Gunnel. 1991b. *Making maximal use of surface criteria in large-scale parsing: the MorP parser*. In PILUS 60, Institute of Linguistics, Stockholm University.

Källgren, Gunnel. 1991c. *Parsing without lexicon: the MorP System*. European Chapter of the Association for Computational Linguistics, Berlin.

Källgren, Gunnel, 1984. *Automatisk excerpering av substantiv ur löpande text. Ett möjligt hjälpmedel vid datoriserad indexering?* Institutet för Rättsinformatik, Juridiska institutionen, Stockholm University.

Magnberg, Sune. 1991. *A Rule-Based System for Identifying Major Syntactic Constituents in Swedish*. Department of Linguistics, Stockholm University.

Malkior, Sven and Carlvik, Mats. 1990. *PC Beta Reference*. Department of Linguistics, Stockholm University.

Marshall, Ian. 1987. *Tag selection using probabilistic methods*. In Garside, Leech and Sampson (1987), chapter 4.

Leech, Geoffrey. 1991. *Corpora and Theories of Linguistic Performance*. Preprint paper for the Nobel Symposium on Corpus Linguistics, Stockholm, 4–8 Augusti 1991.

Lennon, John. *Give Peace A Chance*. Northern Songs, 1969.

Liberman, Mark Y. and Church, Kenneth W. 1992. *Text Analysis and Word Pronunciation in Text-to-Speech Synthesis*. In S. Furui and M. M. Sundhi (eds.), *Advances in Speech Technology*. Marcel Dekker, New York.

Linell, Per. 1972. *Remarks on Swedish Morphology*. Ruul 1, Department of Linguistics, Uppsala University.

Linell, Per. 1976. *On the Structure of Morphological Operations*. In LINGUISTISCHE BERICHTE 44/76, pp. 1–29.

Macintosh Allegro Common LISP 1.3 Reference. Apple Computer, Inc.

Pearl LISP for the Macintosh. Apple Computer, Inc. 1988-89.

Pitkänen, Kari. 1992. *SWETWOL / Major Changes in 1992*. Research Unit for Computational Linguistics, University of Helsinki.

Rankin, Ivan. 1986. *SMORF – an Implementation of Hellberg's Morphology System*. Department of Computer and Information Science, Linköping University.

Sampson, Geoffrey. 1991. *Probabilistic Parsing*. In Svartvik, Jan (ed.), *Directions in Corpus Linguistics 82, Stockholm, 4-8 August 1991*, Mouton de Gruyter, Berlin..

Steele, Guy. 1984. *Common Lisp: The Language*. Digital Press, Bedford, Massachusetts.

Tatar, Deborah. 1987. *A Programmer's Guide to Common Lisp*. Digital Press, Bedford, Massachusetts.

Thorell, Olof. 1973. *Svensk grammatik*. Esselte Studium AB, Stockholm.

Weaver, Warren. 1949. *Translation, I. Machine Translation of Languages*. MIT Press, Cambridge 1955.

APPENDIX A - SOURCE CODE

```
;*****************************************************************
; This program was used to obtain a reversed version of NFO, i.e.
; a version listed according to alphabetically ordered endings.
; The sorting proper was carried out in UNIX.
;*****************************************************************

;;; PROCESS-FILE reverses word tokens in NFO.

(defun process-file (infile outfile)
  (with-open-file (output-stream outfile :direction :output
                                 :if-does-not-exist :create)
    (with-open-file (input-stream infile :direction :input
                                  :if-does-not-exist :error)
      (do* ((first (read input-stream nil nil)
                   (read input-stream nil nil))
            (rest-of-line (read-line input-stream nil nil)
                          (read-line input-stream nil nil)))
           ((null first) 'done!)
        (format output-stream "~A~A~A ~A~%" #\"
                (reverse first) #\" rest-of-line)))))


;*****************************************************************
; These programs were used to initiate hash-tables and hash-arrays
; from the ending lexica text files.
;*****************************************************************

;;; INIT-HASHLEX takes as argument a list of one or more files, and
;;; turns the file(s) into a (i.e., one!) hash-table.
;;; A call looks thus:
;;; ? (init-hashlex '("HD:1" "HD:2" "HD:n"))

(defun init-hashlex (files)
  (setq *hashlex*
        (let ((hash-lexicon (make-hash-table :test #'equal)))
          (dolist (file files)
            (with-open-file (input-stream file :direction :input)
              (do ((entry (read input-stream nil nil)
                          (read input-stream nil nil)))
                  ((null entry) 'done)
                (setf (gethash (car entry) hash-lexicon)
                      (cdr entry)))))
          hash-lexicon)))


;;; TURN-INTO-HASH-ARRAY turns a text file, indicated by its pathname,
;;; into a hash-table. A call looks thus:
;;; ? (turn-into-hash-array "HD:1")

(defun turn-into-hash-array (file)
  (let ((hash-lexicon (make-hash-table :test #'equal)))
    (with-open-file (input-stream file :direction :input)
      (do ((entry (read input-stream nil nil)
                  (read input-stream nil nil)))
          ((null entry) 'done!)
        (setf (gethash (car entry) hash-lexicon) (cdr entry))))
    hash-lexicon))
```

```
;;; INIT-HASH-ARRAY takes a list of files as argument and turns the
;;; files into a vector of hash-tables. A call looks thus:
;;; ? (init-hash-array '("HD:1" "HD:2" "HD:n"))

(defun init-hash-array (list-of-files)
  (setq *hash-array*
        (vector (mapcar #'turn-into-hash-array list-of-files))))


;*****************************************************************
; These programs were used to obtain the ending lexica employed in
; the pattern matching lookup.
;*****************************************************************

;;; GET-ENDINGS takes three arguments. The first is a number n
;;; indicating the number of letters in the ending. The second
;;; argument is the reversed version of the NFO. The third argument
;;; is an outfile with all existing endings of n letters' length
;;; found in NFO, and information about the word classes associated
;;; with the respective endings and each of the word classes'
;;; relative frequency.

(defun get-endings (n infile outfile)
  (with-open-file (output-stream outfile
                                 :direction :output
                                 :if-does-not-exist :create)
    (with-open-file (input-stream infile :direction :input)
      (do ((stop nil) (entry nil))
          (stop (write-entry output-stream entry) 'done!)
        (let* ((word (read file nil nil))
               (freq
                (do ((x (read input-stream nil nil)
                        (read input-stream nil nil)))
                    ((or (numberp x) (null x)) x)))
               (dummy1 (read input-stream nil nil))
               (dummy2 (read input-stream nil nil))
               (wclass (valid-wclass input-stream))
               (dummy3 (read-line input-stream nil nil)))
          (if (null word)
            (setf stop t)
            (if (>= (length word) n)
              (let ((ending (subseq word (- (length word) n))))
                (cond ((null entry)
                       (setf entry
                             `(,ending ,freq ((,wclass ,freq)))))
                      ((string= ending (car entry))
                       (let* ((alist (caddr entry))
                              (wcentry
                               (assoc wclass alist :test #'equal)))
                         (if (null wcentry)
                           (setf alist
                                 (acons wclass `(,freq) alist))
                           (setf (cdr wcentry)
                                 `(,(+ (cadr wcentry) freq))))
                         (setf (cdr entry)
                               `(,(+ (cadr entry) freq) ,alist))))
                      (t (write-entry output-stream entry)
                         (setf entry
                               `(,ending ,freq ((,wclass ,freq))))))))
            ))))))
```

```
;;; VALID-WCLASS checks whether the item found in the word class
;;; position in NFO actually is a word class. If not, the tag
;;; "NT" - for 'Not Tagged (in NFO)' - is returned.

(defun valid-wclass (file)
  (let* ((item (read file nil nil)))
    (if (member item '("**" "ab" "al" "an" "av" "ie" "in"
                       "kn" "nl" "nn" "pm" "pn" "pp" "vb")
                :test #'equal)
      item
      "NT")))


;;; WRITE-ENTRY writes the accumulated ending entries to the
;;; output-stream file.

(defun write-entry (output-stream entry)
  (if (null entry) 'done!
      (format output-stream "~S ~%" (percentage entry))))


;;; PERCENTAGE transforms the word class frequencies from rational
;;; into real numbers.

(defun percentage (list)
  `(,(car list)
    ,(mapcar #'(lambda (s)
                 `(,(car s)
                   ,(float (/ (round (* 10000
                                        (/ (cadr s)
                                           (cadr list))))
                              10000)) ))
             (caddr list))))


;;; SORT-WCLISTS takes a file of ending entries as input and outputs
;;; a file where all the word class lists are sorted according to the
;;; word classes' frequency (in falling order).

(defun sort-wclists (infile outfile)
  (with-open-file (input-stream infile :direction :input)
    (with-open-file (output-stream outfile :direction :output
                                   :if-does-not-exist :create)
      (do* ((entry (read input-stream nil nil)
                   (read input-stream nil nil))
            (ending (car entry) (car entry)))
           ((null entry) 'done!)
        (format output-stream "~S~%"
                (cons ending
                      (list
                       (sort (cadr entry) #'> :key #'cadr))))))))
```

```
;*********************************************************************
; These programs constitute the main programs in the implementation
; of the tagger module.
;*********************************************************************

;;; LOOKUP takes as argument a file of untagged words, iteratively
;;; strips the words of their left-most letter(s), searches for the
;;; remaining 'words' in a dictionary , and, if the dictionary lookup
;;; fails, in the appropriate ending lexicon. The comment ';SUB!' in
;;; the code indicates where one has to decide which subroutine to
;;; employ for the ending lexica lookup. There are three alternatives:
;;; 1. SEARCH-ENDING-LEXICON searches for the endings in text file
;;;    format ending lexica. An ending with n letters is searched in
;;;    the lexicon containing endings with n letters.
;;; 2. SEARCH-HASH-TABLE searches the endings in a hash-table
;;;    containing all endings, irrespective of their length
;;;    (starting with the maximum ending length).
;;; 3. SEARCH-HASH-ARRAY searches for the endings in an array of hash
;;;    tables. An ending with n letters is searched in the hash-table
;;;    where the key (i.e., ending) has n letters.

(defun lookup (infile outfile)
  (with-open-file (output-stream outfile :direction :output
                                 :if-does-not-exist :create)
    (with-open-file (input-stream infile :direction :input)
      (do ((word (get-word input-stream) (get-word input-stream)))
          ((null word) 'done!)
        (do ((rest-of-word word (subseq rest-of-word 1))
             (found nil found))
            ((or found (string= rest-of-word ""))
             (if (not found)
               (format output-stream "(~S ((~S 0.0)))~%" word "NONE"))
             'done!)
          (let* ((info (search-dictionary rest-of-word)))
            (if (null info)
              (setf info (search-ending-lexicon rest-of-word))) ;SUB!
            (if info
              (progn
                (setf found t )
                (format output-stream "~S~%" (cons word info)))))))
      )))


;;; SEARCH-ENDING-LEXIXON searches for an ending of n letters' length
;;; in the ending-lexicon with endings of n letters' length. The
;;; array '"HD:1"...' (and  so forth) describes the current path
;;; in the computer. The figure 7 in the '(> ending-length 7)'
;;; denotes maximal ending length (and thus also number of lexica).

(defun search-ending-lexicon (word)
  (let ((ending-length (length word)))
    (if (or (= ending-length 0) (> ending-length 7))  ;MAX LENGTH
      nil
      (with-open-file
        (infile (aref '#("HD:1" "HD:2" "HD:3" "HD:4"  ;TEXT FILE
                         "HD:5" "HD:6" "HD:7")        ;LEXICA
                      (- ending-length 1))
                :direction :input)
        (do ((text (read infile nil nil) (read infile nil nil))
             (info nil))
            ((or info (null text)) info)
          (if (string= (car text) word)
            (setf info (cdr text)))))))))
```

```
;;; SEARCH-HASH-TABLE searches for the endings in a hash-table. This
;;; program searches the endings in one hash-table only, *HASHLEX*,
;;; which is initiated by the function INIT-HASHLEX.

(defun search-hash-table (word)
  (let ((ending-length (length word)))
    (if (or (= ending-length 0) (> ending-length 7)) ;MAX LENGTH
      nil
      (gethash word *hashlex*))))


;;; SEARCH-HASH-ARRAY searches for the endings in a vector of hash-
;;; tables. An ending with n letters is searched in the hash-table
;;; where the key (i.e., ending) has n letters. the vector of hash-
;;; tables, the global variable *HASH-ARRAY*, is initiated by the
;;; function INIT-HASH-ARRAY.

(defun search-hash-array (word)
  (let ((ending-length (length word)))
    (if (or (= ending-length 0) (> ending-length 7)) ;MAX LENGTH
      nil
      (gethash word
        (aref *hash-array* (- ending-length 1))))))


;;; SEARCH-DICTIONARY searches for a 'stripped' word in a dictionary
;;; (e.g. swetwol). It is here set to nil, not being employed.

(defun search-dictionary (word)
  nil)


;*********************************************************************
; These programs make necessary format transformations.
;*********************************************************************

;;; GET-WORD changes the format of the words in  untagged swetwol
;;; input files into NFO format.

(defun get-word (input-stream)
  (let ((word (read input-stream nil nil)))
    (if (not (null word))
        (progn
          (setf word (cancel< (car word)))
          (setf word (cancel> word))))
    word))


;;; CANCEL> takes away the '>' from the words in the input files of
;;; untagged words.

(defun cancel> (word)
  (setf position (1+ (search ">" (reverse word))))
  (setf word (reverse (subseq (reverse word) position))))


;;; CANCEL< takes away the '<' from the words in the input files of
;;; untagged words.

(defun cancel< (word)
  (setf position (1+ (search "<" word)))
  (setf word (subseq word position)))
```

```
;*********************************************************************
; These programs were used for various counting tasks.
;*********************************************************************

;;; COUNT-ALL takes a text file as argument and counts the number of
;;; words, lines, letters and average word length, and writes the
;;; results to the screen. It is written as a general-purpose program
;;; (instead of writing a program tailor-made for swetwol files).
;;; Because of the swetwol file format,the returned number of lines
;;; was used to read the number of words, since each new word
;;; in untagged swetwol files appears on a specific line.

(defun count-all (filename)
  (with-open-file (infile filename :direction :input)
    (do* ((words 0 words)
          (lines 0 lines)
          (letters 0 letters)
          (chars 0 (+ chars 1))
          (character (read-char infile nil nil)
                     (read-char infile nil nil))
          (state 'blanks state))
         ((null character)
          (if (eq state 'words)
            (progn
              (terpri)
              (setq words (+ words 1))))
          (format t "Lines: ~A~%Words: ~A~%Characters: ~A~%"
                  lines words chars)
          (format t "Letters: ~A~" letters)
          (format t "Average word length: ~2F~" (/ letters words))
          nil)
      (cond
       ((letter? character)
        (setq letters (+ letters 1))
        (setq state 'words))
       (t
        (if (= (char-code #\Newline) (char-code character))
          (setq lines (+ lines 1)))
        (if (eq state 'words)
          (setq words (+ words 1)))
        (setq state 'blanks))))))


;;; LETTER? checks whether a read character is a letter (according to
;;; current definitions!).

(defun letter? (character)
  (let* ((char (char-code (char-downcase character))))
    (or (<= (char-code #\a) char (char-code #\z))
        (= char (char-code #\å))
        (= char (char-code #\ä))
        (= char (char-code #\ö)))))
```

```
;;; COUNT-WORD-CLASSES counts the entries in an ending lexicon and the
;;; frequency of each number of the word classes. The said frequency
;;; is then divided by the number of entries, and thus each word
;;; class' relative frequency is obtained. The number 15 denotes the
;;; number of word class tags currently employed.

(defun count-word-classes (infile)
  (with-open-file (input-stream infile :direction :input)
    (do ((stat (make-array 15 :initial-element 0))
         (entries 0 (+ entries 1))
         (len (length (cadr (read input-stream nil nil)))
              (length (cadr (read input-stream nil nil)))))
        ((zerop len)
         (do ((i 0 (+ i 1)))
             ((> i 14) stat)
           (setf (aref stat i)
                 (float (/ (round (* 1000 (/ (aref stat i) entries)))
                           1000)))))
      (setf (aref stat (- len 1)) (+ 1 (aref stat (- len 1)))))))



;*********************************************************************
; The following programs were not used, but nevertheless implemented
; since their potential function is discussed in the paper.
;*********************************************************************

;;; SPLIT- takes a hyphen compound word entry as argument and returns
;;; the last word of the compound, irrespective of the number of
;;; hyphens used in the compound. Thus:
;;; ? (split- '("jag-vet-inte-vad" (("nn") ("an"))))
;;; "vad"

(defun split- (word-list-entry)
  (setf position (search "-" (reverse (car word-list-entry))))
  (setf word (subseq (car word-list-entry)
                     (- (length (car word-list-entry)) position))))


;;; SPLIT/ takes a slash compound word entry as argument and returns
;;; the last word of the compound. Thus:
;;; ? (split/ '("och/eller" (("nn") ("an"))))
;;; "eller"

(defun split/ (word-list-entry)
  (setf position (1+ (search "/" (car word-list-entry))))
  (setf word (subseq (car word-list-entry) position)))
```

COROLLARY – PROCESS SPEED

The speed at which the module (the `lookup` program) tagged words was controlled – with the predefined LISP function `time` – on one file: ADOP of 191 words. Three different versions of `lookup` were timed. One version of `lookup` used the function `search-ending-lexicon`, which reads from text files. Another version of `lookup` employed the function `search-hashlex`, which searches a hash-table. The third version used the `search-hash-array`, which searches an array of hash-tables, equivalent to the array of text files used by `search-ending-lexicon`. These three program versions were used on one- to four-letter ending lexica. The results are presented in TABLE 8. The results are given in computer ticks rather than seconds, since not *actual*, but *relative* speed is what is of interest here. Moreover, actual speed depends on the hardware employed. All time spent in *Garbage Collection* is subtracted from the figures.

TABLE 8 - Relative process speed given in ticks (Garbage Collection subtracted).

	Program version		
Number of lexica (= letters in endings)	text file lexica	hash-table	hash-table array
One	32 192	5 112	7 580
Two	171 053	3 963	4 830
Three	771 050	2 938	2 936
Four	2 351 926	2 374	2 394

As can be seen, hash-table lookup is not only faster, also, whereas the time required to run the program increases with the number of lexica (= letters in endings) in the text file lookup, it actually *decreases* with an increasing number of ending lexica when hash-tables are used. Thus, when one lexicon is used, hash-table lookup is 6.3 times faster than a text file lookup. When two lexica are used, hash-table lookup is 43 times faster. When three and four lexica are used, the hash-table version of the program runs 262 and 991 times faster, respectively. A dramatic time gain, to euphemize!

The explanation for this is of course that in a text file lookup the lexica will have to be read through until a match is found, and because of the increasing length of the lexica, the process takes more time where more (and increasingly longer) lexica are employed. A hash-table lookup is not dependent on the size of the table (here, ending lexicon). That is to say, a lookup in the one-letter ending lexicon does not take more time than a lookup in the six-letter ending lexicon. This means that what takes time in the hash-table versions of `lookup` is the iterative left-most letter stripping process. The sooner a match can be found, i.e., the more 'to the left' one is in a word when a match is found, the less time will be spent looking for the desired match.

There are, however, some problems associated with hash-tables. First, they require RAM in proportion to their size. It was possible to make a hash-table (or hash-table-array) out of ending lexica one to four on a computer with 4 Mb RAM. However, it was *not* possible to initiate a hash-table using ending lexica one to *seven* on a computer having 8 Mb RAM. Ending lexica one to seven fall a little short of 130 000 entries (lines), which naturally requires a computer with either great RAM capacity, or virtual memory. Thus, using text files might relieve computers with limited RAM (but big hard disks!). Another problem is that the initiation of hash-tables *per se* takes time, and the hash-tables are not saved at computer shut down. Moreover, each time one wants to add a new entry, the hash-table will have to be recalculated

in extenso. Either way, the ending lexica have to be stored as text files on the computer's hard disk, so as to make possible the initiation of the hash-tables.

As can be seen in TABLE 8, the use of one hash-table only is slightly faster than the use of a hash-array. This is due to the fact that in the hash-table version only the ending will have to be searched for, whilst in the hash-array version not only will the ending have to be searched for, but also the appropriate hash-table in the array. However, the time required for calculating the hash-tables is somewhat less for the hash-array version, as can be seen in TABLE 9. The hash-table and hash-table array are initiated by the programs `init-hashlex` and `init-hash-array`, respectively (cf. Appendix A).

Table 9 - Hash table initiation time in ticks (Garbage Collection subtracted).

	Program version	
Number of lexica (= letters in endings)	hash-table	Hash-table array
one	245	223
two	1 896	1 871
three	9 908	9 466
four	47 864	38 737

The time gain is but marginal, when one considers the time saved as compared with the text file lookup, but could be of interest if the ending lexica are extensive.

It must be borne in mind that the test runs accounted for here did not involve any dictionary lookups, which might have found a word prior to the ending lexica being used Thus, each word in this test run was stripped down in an iterative loop to the four-letter length where the search of the ending lexica commenced. These iterative loops were of course pointless, given the module's present stand-alone status, since those lengths greater than four letters were not used to search a dictionary.

The timing of `get-endings` (and other programs) was waived, not being of any relevance to the module.

APPENDIX B - UNTAGGED INFILE

Here an example of an untagged residual file is given. The format is here Times 9 points, whereas the files read by the module were naturally in ASCII format. The file ADOP was chosen because of its modest length. Swedish letters *å*, *ä* and *ö* have been substituted for SWETWOL character representation.

ADOP.UNTAGGED.

```
 ("<*psykisk>")
("<7>")
("<>")
("<*a*c:s>")
("<*a*c>")
("<*afro-*art>")
("<*allers>")
("<*aniana>")
("<*b*u*p>")
("<*b*v*c-mottagning>")
("<*b*v*c>")
("<*bogota>")
("<*bowlby>")
("<*bra-känsla>")
("<*b~hler>")
("<*calcutta>")
("<*d.v.s.>")
("<*dipika>")
("<*djursholms-*bromma-*lidingö-gängen>")
("<*ecuador>")
("<*en-samhet>")
("<*gibran>")
("<*hampstead>")
("<*harlow>")
("<*harlows>")
("<*i*n*t*e-*j*a*g>")
("<*jonna>")
("<*josefin>")
("<*juan-komplex>")
("<*kajsa>")
("<*khalil>")
("<*kittyböcker>")
("<*m*b*d-barn>")
("<*m*b*d-klass>")
("<*m*b*d>")
("<*maharaj>")
("<*mahler>")
("<*malin>")
("<*mamma-och-barn-paret>")
("<*marco>")
("<*micke>")
("<*nio--tioåringen>")
("<*nurseries>")
("<*onej>")
("<*p*b*u>")
("<*peppe>")
("<*petra>")
("<*s*l-kortet>")
("<*s*l>")
("<*singh>")
("<*spitz>")
("<*sri>")
("<*sydindien>")
("<*tove>")
("<*tummelisa>")
("<*utespring>")
("<*winnicott>")
```

("<*åsa>")
("<--*tja>")
("<0--1>")
("<0--2>")
("<0--3>")
("<10--11>")
("<10--15>")
("<11--12>")
("<12--13-årsåldern>")
("<14--15-åringen>")
("<14-månaders>")
("<15--16-åring>")
("<15--16-årsåldern>")
("<19--20>")
("<2--3>")
("<2-3->")
("<22--23>")
("<3--6>")
("<30->")
("<4--5--6-årsåldern>")
("<4-månaders>")
("<40->")
("<5--10>")
("<5--6>")
("<6--7-årsåldern>")
("<6--9>")
("<68-rörelsen>")
("<8-månaders>")
("<9--10-åring>")
("<9--10-åringar>")
("<9--10-årsåldern>")
("<9--10>")
("<9-10->")
("<???>")
("<adoptivskapet>")
("<afro-flätor>")
("<ajajaj>")
("<avladdning>")
("<avudnsjuk>")
("<baby-på-nytt>")
("<baby-test>")
("<barn-på-nytt>")
("<barnpsyk>")
("<bebin>")
("<borderlineproblematik>")
("<brain>")
("<både-god-och-ond>")
("<chokartat>")
("<datafreak>")
("<diarre>")
("<disfunction>")
("<du-och-jag-ensamma-i-världen>")
("<du-och-jag-tillståndet>")
("<du-och-jag>")
("<en-samhet>")
("<en-samt>")
("<en-till-en-relationer>")
("<fascinderad>")
("<förursätter>")
("<göra-få>")
("<ha-galna>")
("<hand-mamman>")
("<hemma-barn>")
("<hjälp-jag>")
("<hospitalism>")
("<hungrig-mat-mätt>")
("<hurpass>")
("<händer-ansikte-röst>")
("<ideer>")

("<igår-idag-imorgon>")
("<inseminationer>")
("<jag-funktioner>")
("<jag-gränser>")
("<jag-kan-inte>")
("<jag-kan-själv>")
("<jag-vill-inte>")
("<jag-vill>")
("<jagochdu>")
("<jagvetintevad>")
("<jätte-macho>")
("<killkompis>")
("<knä-mamman>")
("<la>")
("<lif>")
("<ljuger!>")
("<mamma-ansikten>")
("<mamma-bitarna>")
("<mamma-brister>")
("<mamma-figur>")
("<mamma-händer>")
("<mamma-knän>")
("<mamma-liknande>")
("<mamma-och-barn-par>")
("<mamma-och-pappa-och-barn-triangeln>")
("<mamma-och-pappa-paret>")
("<mamma-pappa-syskon-omgivning>")
("<mammammamm>")
("<mäniska>")
("<nuen>")
("<nyfödd-på-nytt>")
("<odds.>")
("<oefterlängtade>")
("<omgivningoch-barn-par>")
("<ord-symboler>")
("<outhärdiga>")
("<oälskad>")
("<oälskade>")
("<papan>")
("<pappa-och-barnparet>")
("<prata-nivå>")
("<rasta-grejor>")
("<rastafari-gängen>")
("<rastamössorna>")
("<s.k.>")
("<sambande>")
("<sandlådebarnen>")
("<själjsliga>")
("<snuttetrasan>")
("<större.>")
("<suicidförsök>")
("<svagströms-mamma>")
("<svart-vitt>")
("<så.>")
("<t.o.m.>")
("<tittutlekar>")
("<tre-samhet>")
("<två-samhet>")
("<tvåsam-tresam-ensam>")
("<upp-och-ner>")
("<uppvarvning>")
("<urkass>")
("<varken-jag-eller-du>")
("<visa-nivå>")
("<värstingar>")

APPENDIX C - TAGGED OUTFILE

Here an example of a tagged file is given. The format is here Times 9 points, whereas the files read by the module naturally were in ASCII format. Swedish letters *å*, *ä* and *ö* have been substituted for SWETWOL character representation. A listing of the tags is found in § 1.1 (p. 9).

ADOP.TAGGED.

("*psykisk" (("av" 0.922) ("nn" 0.0655) ("pm" 0.0059) ("an" 0.004) ("**" 0.0022) ("ab" 0.0003)))
("7" (("an" 1.0)))
("" (((NONE 0.0)))
("*a*c:s" (("an" 0.8516) ("nl" 0.0801) ("nn" 0.0386) ("pm" 0.0297)))
("*a*c" (("an" 0.6279) ("pm" 0.2869) ("**" 0.0607) ("nn" 0.023) ("NT" 0.0016)))
("*afro-*art" (("av" 0.3111) ("ab" 0.3089) ("vb" 0.1446) ("nn" 0.1072) ("pm" 0.063) ("pn" 0.0575) ("**" 0.0078)))
("*allers" (("nn" 0.392) ("pm" 0.265) ("pn" 0.1123) ("vb" 0.0956) ("ab" 0.0808) ("**" 0.0303) ("an" 0.0171) ("av" 0.007)))
("*aniana" (("nn" 0.5513) ("pn" 0.1789) ("vb" 0.1359) ("av" 0.0897) ("pm" 0.0312) ("ab" 0.0117) ("**" 0.0008) ("an" 0.0003) ("NT" 0.0002) ("al" 0.0)))
("*b*u*p" (("nn" 0.5476) ("ab" 0.3308) ("pm" 0.0542) ("an" 0.026) ("av" 0.0143) ("vb" 0.0123) ("**" 0.0114) ("pn" 0.0025) ("in" 0.0007) ("NT" 0.0002)))
("*b*v*c-mottagning" (("nn" 0.8346) ("pm" 0.0343) ("av" 0.0315) ("pp" 0.031) ("pn" 0.0296) ("ab" 0.0233) ("**" 0.0108) ("vb" 0.0039) ("NT" 0.0005) ("an" 0.0002) ("in" 0.0002)))
("*b*v*c" (("an" 0.6279) ("pm" 0.2869) ("**" 0.0607) ("nn" 0.023) ("NT" 0.0016)))
("*bogota" (("av" 0.3081) ("vb" 0.2998) ("pn" 0.1718) ("nl" 0.0788) ("nn" 0.0528) ("ab" 0.0523) ("pm" 0.0301) ("**" 0.0042) ("an" 0.002) ("NT" 0.0002)))
("*bowlby" (("pm" 0.6683) ("nn" 0.2439) ("**" 0.0829) ("NT" 0.0049)))
("*bra-känsla" (("av" 0.4597) ("vb" 0.2149) ("pn" 0.1864) ("nn" 0.0726) ("pm" 0.0389) ("ab" 0.0141) ("**" 0.0129) ("NT" 0.0003) ("in" 0.0001)))
("*b~hler" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab" 0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018) ("NT" 0.0003)))
("*calcutta" (("av" 0.3081) ("vb" 0.2998) ("pn" 0.1718) ("nl" 0.0788) ("nn" 0.0528) ("ab" 0.0523) ("pm" 0.0301) ("**" 0.0042) ("an" 0.002) ("NT" 0.0002)))
("*d.v.s." (("an" 1.0)))
("*dipika" (("av" 0.6322) ("vb" 0.1411) ("ab" 0.0956) ("nn" 0.074) ("pn" 0.0344) ("pm" 0.0227)))
("*djursholms-*bromma-*lidingö-gängen" (("nn" 0.4208) ("al" 0.3034) ("pn" 0.0738) ("kn" 0.0648) ("ab" 0.0525) ("pm" 0.0356) ("nl" 0.024) ("av" 0.0173) ("vb" 0.0054) ("**" 0.002) ("NT" 0.0002) ("pp" 0.0001) ("in" 0.0)))
("*ecuador" (("nn" 0.7478) ("av" 0.1217) ("vb" 0.0983) ("pm" 0.0221) ("**" 0.01) ("NT" 0.0002)))
("*en-samhet" (("nn" 0.4425) ("pn" 0.3612) ("al" 0.0977) ("ab" 0.045) ("vb" 0.021) ("pm" 0.0187) ("av" 0.0121) ("**" 0.0017) ("an" 0.0001) ("NT" 0.0001)))
("*gibran" (("pn" 0.4237) ("nn" 0.1953) ("vb" 0.1296) ("ab" 0.0795) ("pp" 0.0772) ("kn" 0.049) ("pm" 0.0379) ("**" 0.0051) ("av" 0.0021) ("in" 0.0002) ("nl" 0.0002) ("NT" 0.0002) ("an" 0.0001)))
("*hampstead" (("vb" 0.3732) ("pn" 0.2646) ("nn" 0.233) ("av" 0.1022) ("pm" 0.0253) ("**" 0.0009) ("NT" 0.0005) ("ab" 0.0003)))
("*harlow" (("pm" 0.5821) ("**" 0.2836) ("nn" 0.1343)))
("*harlows" (("pm" 0.6786) ("**" 0.2857) ("nn" 0.0357)))
("*i*n*t*e-*j*a*g" (("nn" 0.4509) ("pn" 0.2639) ("av" 0.1329) ("pm" 0.0483) ("ab" 0.0444) ("vb" 0.0373) ("pp" 0.0119) ("**" 0.005) ("an" 0.0047) ("NT" 0.0005) ("in" 0.0002)))
("*jonna" (("nn" 0.5513) ("pn" 0.1789) ("vb" 0.1359) ("av" 0.0897) ("pm" 0.0312) ("ab" 0.0117) ("**" 0.0008) ("an" 0.0003) ("NT" 0.0002) ("al" 0.0)))
("*josefin" (("pn" 0.5181) ("ab" 0.1957) ("pm" 0.1291) ("nn" 0.126) ("av" 0.0152) ("**" 0.0152) ("an" 0.0008)))
("*juan-komplex" (("an" 0.598) ("nl" 0.2606) ("nn" 0.1005) ("pm" 0.017) ("**" 0.0119) ("NT" 0.0068) ("av" 0.0051)))
("*kajsa" (("pn" 0.2977) ("vb" 0.2439) ("av" 0.2401) ("nn" 0.0977) ("an" 0.0955) ("pm" 0.0216) ("**" 0.0033) ("NT" 0.0003)))
("*khalil" (("nn" 0.7465) ("pm" 0.1282) ("av" 0.0535) ("**" 0.038) ("an" 0.031) ("NT" 0.0028)))
("*kittyböcker" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab" 0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018) ("NT" 0.0003)))
("*m*b*d-barn" (("nn" 0.8012) ("pm" 0.1299) ("av" 0.0615) ("**" 0.0056) ("NT" 0.0019)))
("*m*b*d-klass" (("pn" 0.6005) ("nn" 0.1549) ("av" 0.1208) ("ab" 0.0473) ("pm" 0.0453) ("**" 0.0166) ("vb" 0.0104) ("an" 0.0037) ("in" 0.0004)))
("*m*b*d" (("pp" 0.4036) ("nn" 0.2392) ("vb" 0.1063) ("pm" 0.0762) ("ab" 0.0665) ("av" 0.0494) ("pn" 0.0453) ("**" 0.0073) ("an" 0.0057) ("NT" 0.0003) ("in" 0.0001) ("kn" 0.0001)))

("*maharaj" (("nn" 0.8273) ("pm" 0.1364) ("NT" 0.0273) ("in" 0.0091)))
("*mahler" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab" 0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018) ("NT" 0.0003)))
("*malin" (("pn" 0.5181) ("ab" 0.1957) ("pm" 0.1291) ("nn" 0.126) ("av" 0.0152) ("**" 0.0152) ("an" 0.0008)))
("*mamma-och-barn-paret" (("nn" 0.4425) ("pn" 0.3612) ("al" 0.0977) ("ab" 0.045) ("vb" 0.021) ("pm" 0.0187) ("av" 0.0121) ("**" 0.0017) ("an" 0.0001) ("NT" 0.0001)))
("*marco" (("pm" 0.808) ("an" 0.112) ("**" 0.064) ("nn" 0.016)))
("*micke" (("ab" 0.4687) ("nn" 0.1829) ("av" 0.1555) ("pm" 0.1548) ("vb" 0.0324) ("**" 0.005) ("NT" 0.0007)))
("*nio--tioåringen" (("nn" 0.4208) ("al" 0.3034) ("pn" 0.0738) ("kn" 0.0648) ("ab" 0.0525) ("pm" 0.0356) ("nl" 0.024) ("av" 0.0173) ("vb" 0.0054) ("**" 0.002) ("NT" 0.0002) ("pp" 0.0001) ("in" 0.0)))
("*nurseries" (("vb" 0.5831) ("pm" 0.1719) ("ab" 0.0781) ("pn" 0.0669) ("nn" 0.0471) ("**" 0.0386) ("av" 0.0139) ("pp" 0.0003)))
("*onej" (("ab" 0.5553) ("in" 0.2826) ("pn" 0.085) ("nn" 0.0494) ("pm" 0.0277)))
("*p*b*u" (("ab" 0.7944) ("nn" 0.0701) ("pm" 0.047) ("nl" 0.0257) ("**" 0.0257) ("pn" 0.025) ("kn" 0.0058) ("an" 0.0054) ("NT" 0.0009) ("in" 0.0002)))
("*peppe" (("ab" 0.5591) ("pm" 0.2151) ("nn" 0.1452) ("**" 0.0753) ("av" 0.0054)))
("*petra" (("vb" 0.3749) ("pn" 0.2303) ("av" 0.1758) ("ab" 0.1439) ("nl" 0.0407) ("nn" 0.0207) ("pm" 0.0096) ("**" 0.0022) ("kn" 0.0009) ("pp" 0.0005) ("NT" 0.0002) ("in" 0.0001) ("an" 0.0001)))
("*s*l-kortet" (("nn" 0.4425) ("pn" 0.3612) ("al" 0.0977) ("ab" 0.045) ("vb" 0.021) ("pm" 0.0187) ("av" 0.0121) ("**" 0.0017) ("an" 0.0001) ("NT" 0.0001)))
("*s*l" (("pp" 0.351) ("nn" 0.306) ("vb" 0.1163) ("av" 0.0648) ("ab" 0.0618) ("pm" 0.0537) ("an" 0.0184) ("pn" 0.0116) ("**" 0.0101) ("kn" 0.0057) ("in" 0.0002) ("NT" 0.0002) ("nl" 0.0001)))
("*singh" (("pm" 0.6552) ("**" 0.3103) ("nn" 0.0345)))
("*spitz" (("pm" 0.9632) ("nn" 0.0294) ("**" 0.0074)))
("*sri" (("nn" 0.6886) ("av" 0.1546) ("pm" 0.0784) ("ab" 0.0611) ("**" 0.0127) ("an" 0.0046)))
("*sydindien" (("nn" 0.4208) ("al" 0.3034) ("pn" 0.0738) ("kn" 0.0648) ("ab" 0.0525) ("pm" 0.0356) ("nl" 0.024) ("av" 0.0173) ("vb" 0.0054) ("**" 0.002) ("NT" 0.0002) ("pp" 0.0001) ("in" 0.0)))
("*tove" (("pm" 0.3498) ("kn" 0.1486) ("**" 0.1455) ("nn" 0.1084) ("pp" 0.0836) ("av" 0.0619) ("vb" 0.0526) ("pn" 0.0495)))
("*tummelisa" (("pn" 0.2977) ("vb" 0.2439) ("av" 0.2401) ("nn" 0.0977) ("an" 0.0955) ("pm" 0.0216) ("**" 0.0033) ("NT" 0.0003)))
("*utespring" (("nn" 0.8346) ("pm" 0.0343) ("av" 0.0315) ("pp" 0.031) ("pn" 0.0296) ("ab" 0.0233) ("**" 0.0108) ("vb" 0.0039) ("NT" 0.0005) ("an" 0.0002) ("in" 0.0002)))
("*winnicott" (("ie" 0.3099) ("kn" 0.2784) ("al" 0.1904) ("vb" 0.065) ("nn" 0.0538) ("pn" 0.0318) ("av" 0.0279) ("ab" 0.0239) ("nl" 0.0155) ("pm" 0.003) ("NT" 0.0003) ("**" 0.0001) ("an" 0.0001)))
("*åsa" (("pn" 0.2977) ("vb" 0.2439) ("av" 0.2401) ("nn" 0.0977) ("an" 0.0955) ("pm" 0.0216) ("**" 0.0033) ("NT" 0.0003)))
("--*tja" (("vb" 0.6203) ("in" 0.1757) ("nn" 0.1496) ("pm" 0.051) ("**" 0.0034)))
("0--1" (("an" 1.0)))
("0--2" (("an" 1.0)))
("0--3" (("an" 1.0)))
("10--11" (("an" 1.0)))
("10--15" (("an" 1.0)))
("11--12" (("an" 1.0)))
("12--13-årsåldern" (("nn" 0.8012) ("pm" 0.1299) ("av" 0.0615) ("**" 0.0056) ("NT" 0.0019)))
("14--15-åringen" (("nn" 0.4208) ("al" 0.3034) ("pn" 0.0738) ("kn" 0.0648) ("ab" 0.0525) ("pm" 0.0356) ("nl" 0.024) ("av" 0.0173) ("vb" 0.0054) ("**" 0.002) ("NT" 0.0002) ("pp" 0.0001) ("in" 0.0)))
("14-månaders" (("nn" 0.392) ("pm" 0.265) ("pn" 0.1123) ("vb" 0.0956) ("ab" 0.0808) ("**" 0.0303) ("an" 0.0171) ("av" 0.007)))
("15--16-åring" (("nn" 0.8346) ("pm" 0.0343) ("av" 0.0315) ("pp" 0.031) ("pn" 0.0296) ("ab" 0.0233) ("**" 0.0108) ("vb" 0.0039) ("NT" 0.0005) ("an" 0.0002) ("in" 0.0002)))
("15--16-årsåldern" (("nn" 0.8012) ("pm" 0.1299) ("av" 0.0615) ("**" 0.0056) ("NT" 0.0019)))
("19--20" (("an" 1.0)))
("2--3" (("an" 1.0)))
("2-3-" (("NT" 1.0)))
("22--23" (("an" 1.0)))
("3--6" (("an" 1.0)))
("30-" (("NT" 1.0)))
("4--5--6-årsåldern" (("nn" 0.8012) ("pm" 0.1299) ("av" 0.0615) ("**" 0.0056) ("NT" 0.0019)))
("4-månaders" (("nn" 0.392) ("pm" 0.265) ("pn" 0.1123) ("vb" 0.0956) ("ab" 0.0808) ("**" 0.0303) ("an" 0.0171) ("av" 0.007)))
("40-" (("NT" 1.0)))
("5--10" (("an" 1.0)))
("5--6" (("an" 1.0)))
("6--7-årsåldern" (("nn" 0.8012) ("pm" 0.1299) ("av" 0.0615) ("**" 0.0056) ("NT" 0.0019)))
("6--9" (("an" 1.0)))

("68-rörelsen" (("nn" 0.4208) ("al" 0.3034) ("pn" 0.0738) ("kn" 0.0648) ("ab" 0.0525) ("pm" 0.0356) ("nl" 0.024) ("av" 0.0173) ("vb" 0.0054) ("**" 0.002) ("NT" 0.0002) ("pp" 0.0001) ("in" 0.0)))
("8-månaders" (("nn" 0.392) ("pm" 0.265) ("pn" 0.1123) ("vb" 0.0956) ("ab" 0.0808) ("**" 0.0303) ("an" 0.0171) ("av" 0.007)))
("9--10-åring" (("nn" 0.8346) ("pm" 0.0343) ("av" 0.0315) ("pp" 0.031) ("pn" 0.0296) ("ab" 0.0233) ("**" 0.0108) ("vb" 0.0039) ("NT" 0.0005) ("an" 0.0002) ("in" 0.0002)))
("9--10-åringar" (("vb" 0.6909) ("nn" 0.2646) ("pm" 0.0161) ("ab" 0.0113) ("av" 0.0108) ("pn" 0.0051) ("**" 0.0009) ("an" 0.0002) ("NT" 0.0)))
("9--10-årsåldern" (("nn" 0.8012) ("pm" 0.1299) ("av" 0.0615) ("**" 0.0056) ("NT" 0.0019)))
("9--10" (("an" 1.0)))
("9-10-" (("NT" 1.0)))
("???" ((NONE 0.0)))
("adoptivskapet" (("nn" 0.4425) ("pn" 0.3612) ("al" 0.0977) ("ab" 0.045) ("vb" 0.021) ("pm" 0.0187) ("av" 0.0121) ("**" 0.0017) ("an" 0.0001) ("NT" 0.0001)))
("afro-flätor" (("nn" 0.7478) ("av" 0.1217) ("vb" 0.0983) ("pm" 0.0221) ("**" 0.01) ("NT" 0.0002)))
("ajajaj" (("nn" 0.8273) ("pm" 0.1364) ("NT" 0.0273) ("in" 0.0091)))
("avladdning" (("nn" 0.8346) ("pm" 0.0343) ("av" 0.0315) ("pp" 0.031) ("pn" 0.0296) ("ab" 0.0233) ("**" 0.0108) ("vb" 0.0039) ("NT" 0.0005) ("an" 0.0002) ("in" 0.0002)))
("avudnsjuk" (("nn" 0.7452) ("av" 0.2357) ("pm" 0.019)))
("baby-på-nytt" (("ie" 0.3099) ("kn" 0.2784) ("al" 0.1904) ("vb" 0.065) ("nn" 0.0538) ("pn" 0.0318) ("av" 0.0279) ("ab" 0.0239) ("nl" 0.0155) ("pm" 0.003) ("NT" 0.0003) ("**" 0.0001) ("an" 0.0001)))
("baby-test" (("ab" 0.6004) ("nn" 0.2017) ("av" 0.1042) ("pm" 0.0343) ("vb" 0.0257) ("kn" 0.0137) ("**" 0.0111) ("an" 0.0067) ("in" 0.001) ("pp" 0.001) ("NT" 0.0003)))
("barn-på-nytt" (("ie" 0.3099) ("kn" 0.2784) ("al" 0.1904) ("vb" 0.065) ("nn" 0.0538) ("pn" 0.0318) ("av" 0.0279) ("ab" 0.0239) ("nl" 0.0155) ("pm" 0.003) ("NT" 0.0003) ("**" 0.0001) ("an" 0.0001)))
("barnpsyk" (("pm" 0.5455) ("nn" 0.4545)))
("bebin" (("pn" 0.5181) ("ab" 0.1957) ("pm" 0.1291) ("nn" 0.126) ("av" 0.0152) ("**" 0.0152) ("an" 0.0008)))
("borderlineproblematik" (("nn" 0.7484) ("pm" 0.148) ("av" 0.0905) ("pn" 0.0078) ("**" 0.0047) ("NT" 0.0005)))
("brain" (("pn" 0.5181) ("ab" 0.1957) ("pm" 0.1291) ("nn" 0.126) ("av" 0.0152) ("**" 0.0152) ("an" 0.0008)))
("både-god-och-ond" (("nn" 0.4274) ("pm" 0.3019) ("pp" 0.1205) ("ab" 0.0703) ("av" 0.0354) ("**" 0.0282) ("vb" 0.0131) ("an" 0.0018) ("NT" 0.0009) ("kn" 0.0004)))
("chokartat" (("vb" 0.6575) ("nn" 0.149) ("pn" 0.0988) ("av" 0.0478) ("ab" 0.0322) ("pm" 0.0087) ("**" 0.0054) ("nl" 0.0003) ("NT" 0.0001) ("kn" 0.0001)))
("datafreak" (("nn" 0.8612) ("pm" 0.0607) ("av" 0.026) ("an" 0.0217) ("ab" 0.0195) ("pp" 0.0108)))
("diarre" (("av" 0.3383) ("nn" 0.3065) ("ab" 0.2201) ("nl" 0.0516) ("vb" 0.0271) ("pp" 0.0229) ("pm" 0.02) ("**" 0.0064) ("kn" 0.0051) ("pn" 0.0016) ("NT" 0.0005)))
("disfunction" (("nn" 0.4076) ("pn" 0.2924) ("pm" 0.2567) ("**" 0.0318) ("nl" 0.0105) ("av" 0.0005) ("NT" 0.0005) ("in" 0.0001)))
("du-och-jag-ensamma-i-världen" (("nn" 0.4208) ("al" 0.3034) ("pn" 0.0738) ("kn" 0.0648) ("ab" 0.0525) ("pm" 0.0356) ("nl" 0.024) ("av" 0.0173) ("vb" 0.0054) ("**" 0.002) ("NT" 0.0002) ("pp" 0.0001) ("in" 0.0)))
("du-och-jag-tillståndet" (("nn" 0.4425) ("pn" 0.3612) ("al" 0.0977) ("ab" 0.045) ("vb" 0.021) ("pm" 0.0187) ("av" 0.0121) ("**" 0.0017) ("an" 0.0001) ("NT" 0.0001)))
("du-och-jag" (("nn" 0.5004) ("pn" 0.4728) ("ab" 0.0088) ("av" 0.0066) ("pm" 0.0063) ("vb" 0.0024) ("**" 0.0013) ("NT" 0.0006) ("in" 0.0006) ("an" 0.0004)))
("en-samhet" (("nn" 0.4425) ("pn" 0.3612) ("al" 0.0977) ("ab" 0.045) ("vb" 0.021) ("pm" 0.0187) ("av" 0.0121) ("**" 0.0017) ("an" 0.0001) ("NT" 0.0001)))
("en-samt" (("ab" 0.3736) ("kn" 0.2841) ("av" 0.2053) ("vb" 0.1067) ("nn" 0.0263) ("an" 0.0025) ("pm" 0.0008) ("**" 0.0008)))
("en-till-en-relationer" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab" 0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018) ("NT" 0.0003)))
("fascinderad" (("vb" 0.3732) ("pn" 0.2646) ("nn" 0.233) ("av" 0.1022) ("pm" 0.0253) ("**" 0.0009) ("NT" 0.0005) ("ab" 0.0003)))
("förursätter" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab" 0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018) ("NT" 0.0003)))
("göra-få" (("vb" 0.8632) ("av" 0.1353) ("nn" 0.0014)))
("ha-galna" (("nn" 0.5513) ("pn" 0.1789) ("vb" 0.1359) ("av" 0.0897) ("pm" 0.0312) ("ab" 0.0117) ("**" 0.0008) ("an" 0.0003) ("NT" 0.0002) ("al" 0.0)))
("hand-mamman" (("pn" 0.4237) ("nn" 0.1953) ("vb" 0.1296) ("ab" 0.0795) ("pp" 0.0772) ("kn" 0.049) ("pm" 0.0379) ("**" 0.0051) ("av" 0.0021) ("in" 0.0002) ("nl" 0.0002) ("NT" 0.0002) ("an" 0.0001)))
("hemma-barn" (("nn" 0.8012) ("pm" 0.1299) ("av" 0.0615) ("**" 0.0056) ("NT" 0.0019)))
("hjälp-jag" (("nn" 0.5004) ("pn" 0.4728) ("ab" 0.0088) ("av" 0.0066) ("pm" 0.0063) ("vb" 0.0024) ("**" 0.0013) ("NT" 0.0006) ("in" 0.0006) ("an" 0.0004)))
("hospitalism" (("nn" 0.9955) ("an" 0.0045)))
("hungrig-mat-mätt" (("ie" 0.3099) ("kn" 0.2784) ("al" 0.1904) ("vb" 0.065) ("nn" 0.0538) ("pn" 0.0318) ("av" 0.0279) ("ab" 0.0239) ("nl" 0.0155) ("pm" 0.003) ("NT" 0.0003) ("**" 0.0001) ("an" 0.0001)))

("hurpass" (("pn" 0.6005) ("nn" 0.1549) ("av" 0.1208) ("ab" 0.0473) ("pm" 0.0453) ("**" 0.0166) ("vb" 0.0104) ("an" 0.0037) ("in" 0.0004)))
("händer-ansikte-röst" (("ab" 0.6004) ("nn" 0.2017) ("av" 0.1042) ("pm" 0.0343) ("vb" 0.0257) ("kn" 0.0137) ("**" 0.0111) ("an" 0.0067) ("in" 0.001) ("pp" 0.001) ("NT" 0.0003)))
("ideer" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab" 0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018) ("NT" 0.0003)))
("igår-idag-imorgon" (("nn" 0.4076) ("pn" 0.2924) ("pm" 0.2567) ("**" 0.0318) ("nl" 0.0105) ("av" 0.0005) ("NT" 0.0005) ("in" 0.0001)))
("inseminationer" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab" 0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018) ("NT" 0.0003)))
("jag-funktioner" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab" 0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018) ("NT" 0.0003)))
("jag-gränser" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab" 0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018) ("NT" 0.0003)))
("jag-kan-inte" (("ab" 0.6264) ("vb" 0.1703) ("av" 0.096) ("nn" 0.0778) ("pm" 0.0139) ("**" 0.0062) ("pn" 0.0039) ("nl" 0.0033) ("pp" 0.0021) ("NT" 0.0001)))
("jag-kan-själv" (("av" 0.8284) ("pn" 0.0826) ("nl" 0.0551) ("nn" 0.0265) ("an" 0.0032) ("pm" 0.0021) ("**" 0.0011) ("vb" 0.0011)))
("jag-vill-inte" (("ab" 0.6264) ("vb" 0.1703) ("av" 0.096) ("nn" 0.0778) ("pm" 0.0139) ("**" 0.0062) ("pn" 0.0039) ("nl" 0.0033) ("pp" 0.0021) ("NT" 0.0001)))
("jag-vill" (("pp" 0.5629) ("vb" 0.1839) ("nn" 0.1253) ("av" 0.0454) ("ab" 0.0376) ("pm" 0.023) ("pn" 0.018) ("**" 0.0021) ("kn" 0.0014) ("nl" 0.0002) ("an" 0.0001) ("NT" 0.0001) ("in" 0.0001)))
("jagochdu" (("pn" 0.7436) ("**" 0.1987) ("pm" 0.0321) ("an" 0.0256)))
("jagvetintevad" (("vb" 0.3732) ("pn" 0.2646) ("nn" 0.233) ("av" 0.1022) ("pm" 0.0253) ("**" 0.0009) ("NT" 0.0005) ("ab" 0.0003)))
("jätte-macho" (("pm" 0.7317) ("an" 0.1707) ("**" 0.0488) ("nn" 0.0244) ("pn" 0.0244)))
("killkompis" (("ab" 0.4807) ("pm" 0.2589) ("nn" 0.2176) ("**" 0.0231) ("av" 0.0104) ("an" 0.0042) ("pn" 0.0039) ("pp" 0.0012)))
("knä-mamman" (("pn" 0.4237) ("nn" 0.1953) ("vb" 0.1296) ("ab" 0.0795) ("pp" 0.0772) ("kn" 0.049) ("pm" 0.0379) ("**" 0.0051) ("av" 0.0021) ("in" 0.0002) ("nl" 0.0002) ("NT" 0.0002) ("an" 0.0001)))
("la" (("av" 0.4597) ("vb" 0.2149) ("pn" 0.1864) ("nn" 0.0726) ("pm" 0.0389) ("ab" 0.0141) ("**" 0.0129) ("NT" 0.0003) ("in" 0.0001)))
("lif" (("pm" 0.9032) ("**" 0.0968)))
("ljuger!" ((NONE 0.0)))
("mamma-ansikten" (("nn" 0.4208) ("al" 0.3034) ("pn" 0.0738) ("kn" 0.0648) ("ab" 0.0525) ("pm" 0.0356) ("nl" 0.024) ("av" 0.0173) ("vb" 0.0054) ("**" 0.002) ("NT" 0.0002) ("pp" 0.0001) ("in" 0.0)))
("mamma-bitarna" (("nn" 0.5513) ("pn" 0.1789) ("vb" 0.1359) ("av" 0.0897) ("pm" 0.0312) ("ab" 0.0117) ("**" 0.0008) ("an" 0.0003) ("NT" 0.0002) ("al" 0.0)))
("mamma-brister" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab" 0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018) ("NT" 0.0003)))
("mamma-figur" (("ab" 0.4007) ("nn" 0.3089) ("pp" 0.2412) ("pm" 0.0315) ("**" 0.0134) ("av" 0.0034) ("an" 0.001)))
("mamma-händer" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab" 0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018) ("NT" 0.0003)))
("mamma-knän" (("kn" 0.6688) ("nn" 0.1952) ("ab" 0.0915) ("av" 0.0376) ("pm" 0.007)))
("mamma-liknande" (("vb" 0.5279) ("al" 0.1503) ("pn" 0.1319) ("nn" 0.0735) ("av" 0.0682) ("kn" 0.0184) ("ab" 0.0163) ("**" 0.0085) ("pm" 0.0025) ("nl" 0.0023) ("NT" 0.0001) ("an" 0.0001)))
("mamma-och-barn-par" (("vb" 0.6909) ("nn" 0.2646) ("pm" 0.0161) ("ab" 0.0113) ("av" 0.0108) ("pn" 0.0051) ("**" 0.0009) ("an" 0.0002) ("NT" 0.0)))
("mamma-och-pappa-och-barn-triangeln" (("nn" 0.9267) ("pm" 0.0659) ("an" 0.0055) ("**" 0.0018)))
("mamma-och-pappa-paret" (("nn" 0.4425) ("pn" 0.3612) ("al" 0.0977) ("ab" 0.045) ("vb" 0.021) ("pm" 0.0187) ("av" 0.0121) ("**" 0.0017) ("an" 0.0001) ("NT" 0.0001)))
("mamma-pappa-syskon-omgivning" (("nn" 0.8346) ("pm" 0.0343) ("av" 0.0315) ("pp" 0.031) ("pn" 0.0296) ("ab" 0.0233) ("**" 0.0108) ("vb" 0.0039) ("NT" 0.0005) ("an" 0.0002) ("in" 0.0002)))
("mammammamm" (("an" 0.4737) ("nn" 0.2807) ("pm" 0.2105) ("**" 0.0351)))
("mäniska" (("av" 0.6322) ("vb" 0.1411) ("ab" 0.0956) ("nn" 0.074) ("pn" 0.0344) ("pm" 0.0227)))
("nuen" (("nn" 0.4208) ("al" 0.3034) ("pn" 0.0738) ("kn" 0.0648) ("ab" 0.0525) ("pm" 0.0356) ("nl" 0.024) ("av" 0.0173) ("vb" 0.0054) ("**" 0.002) ("NT" 0.0002) ("pp" 0.0001) ("in" 0.0)))
("nyfödd-på-nytt" (("ie" 0.3099) ("kn" 0.2784) ("al" 0.1904) ("vb" 0.065) ("nn" 0.0538) ("pn" 0.0318) ("av" 0.0279) ("ab" 0.0239) ("nl" 0.0155) ("pm" 0.003) ("NT" 0.0003) ("**" 0.0001) ("an" 0.0001)))
("odds." (("an" 1.0)))
("oefterlängtade" (("vb" 0.5279) ("al" 0.1503) ("pn" 0.1319) ("nn" 0.0735) ("av" 0.0682) ("kn" 0.0184) ("ab" 0.0163) ("**" 0.0085) ("pm" 0.0025) ("nl" 0.0023) ("NT" 0.0001) ("an" 0.0001)))
("omgivningoch-barn-par" (("vb" 0.6909) ("nn" 0.2646) ("pm" 0.0161) ("ab" 0.0113) ("av" 0.0108) ("pn" 0.0051) ("**" 0.0009) ("an" 0.0002) ("NT" 0.0)))
("ord-symboler" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab" 0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018) ("NT" 0.0003)))
("outhärdiga" (("av" 0.5724) ("pn" 0.1803) ("vb" 0.1468) ("nn" 0.0778) ("ab" 0.0155) ("pm" 0.007) ("**" 0.0001) ("NT" 0.0001) ("an" 0.0001)))

("oälskad" (("vb" 0.3732) ("pn" 0.2646) ("nn" 0.233) ("av" 0.1022) ("pm" 0.0253) ("**" 0.0009) ("NT" 0.0005) ("ab" 0.0003)))
("oälskade" (("vb" 0.5279) ("al" 0.1503) ("pn" 0.1319) ("nn" 0.0735) ("av" 0.0682) ("kn" 0.0184) ("ab" 0.0163) ("**" 0.0085) ("pm" 0.0025) ("nl" 0.0023) ("NT" 0.0001) ("an" 0.0001)))
("papan" (("pn" 0.4237) ("nn" 0.1953) ("vb" 0.1296) ("ab" 0.0795) ("pp" 0.0772) ("kn" 0.049) ("pm" 0.0379) ("**" 0.0051) ("av" 0.0021) ("in" 0.0002) ("nl" 0.0002) ("NT" 0.0002) ("an" 0.0001)))
("pappa-och-barnparet" (("nn" 0.4425) ("pn" 0.3612) ("al" 0.0977) ("ab" 0.045) ("vb" 0.021) ("pm" 0.0187) ("av" 0.0121) ("**" 0.0017) ("an" 0.0001) ("NT" 0.0001)))
("prata-nivå" (("nl" 0.9198) ("nn" 0.0795) ("vb" 0.0008)))
("rasta-grejor" (("nn" 0.7478) ("av" 0.1217) ("vb" 0.0983) ("pm" 0.0221) ("**" 0.01) ("NT" 0.0002)))
("rastafari-gängen" (("nn" 0.4208) ("al" 0.3034) ("pn" 0.0738) ("kn" 0.0648) ("ab" 0.0525) ("pm" 0.0356) ("nl" 0.024) ("av" 0.0173) ("vb" 0.0054) ("**" 0.002) ("NT" 0.0002) ("pp" 0.0001) ("in" 0.0)))
("rastamössorna" (("nn" 0.5513) ("pn" 0.1789) ("vb" 0.1359) ("av" 0.0897) ("pm" 0.0312) ("ab" 0.0117) ("**" 0.0008) ("an" 0.0003) ("NT" 0.0002) ("al" 0.0)))
("s.k." (("an" 1.0)))
("sambande" (("vb" 0.5279) ("al" 0.1503) ("pn" 0.1319) ("nn" 0.0735) ("av" 0.0682) ("kn" 0.0184) ("ab" 0.0163) ("**" 0.0085) ("pm" 0.0025) ("nl" 0.0023) ("NT" 0.0001) ("an" 0.0001)))
("sandlådebarnen" (("nn" 0.4208) ("al" 0.3034) ("pn" 0.0738) ("kn" 0.0648) ("ab" 0.0525) ("pm" 0.0356) ("nl" 0.024) ("av" 0.0173) ("vb" 0.0054) ("**" 0.002) ("NT" 0.0002) ("pp" 0.0001) ("in" 0.0)))
("själjsliga" (("av" 0.5724) ("pn" 0.1803) ("vb" 0.1468) ("nn" 0.0778) ("ab" 0.0155) ("pm" 0.007) ("**" 0.0001) ("NT" 0.0001) ("an" 0.0001)))
("snuttetrasan" (("pn" 0.4237) ("nn" 0.1953) ("vb" 0.1296) ("ab" 0.0795) ("pp" 0.0772) ("kn" 0.049) ("pm" 0.0379) ("**" 0.0051) ("av" 0.0021) ("in" 0.0002) ("nl" 0.0002) ("NT" 0.0002) ("an" 0.0001)))
("större." (("an" 1.0)))
("suicidförsök" (("nn" 0.8585) ("vb" 0.1185) ("pm" 0.0229)))
("svagströms-mamma" (("vb" 0.3178) ("pn" 0.2944) ("av" 0.1912) ("nn" 0.0975) ("ab" 0.0584) ("pm" 0.035) ("**" 0.0053) ("in" 0.0003)))
("svart-vitt" (("ie" 0.3099) ("kn" 0.2784) ("al" 0.1904) ("vb" 0.065) ("nn" 0.0538) ("pn" 0.0318) ("av" 0.0279) ("ab" 0.0239) ("nl" 0.0155) ("pm" 0.003) ("NT" 0.0003) ("**" 0.0001) ("an" 0.0001)))
("så." (("an" 1.0)))
("t.o.m." (("an" 1.0)))
("tittutlekar" (("vb" 0.6909) ("nn" 0.2646) ("pm" 0.0161) ("ab" 0.0113) ("av" 0.0108) ("pn" 0.0051) ("**" 0.0009) ("an" 0.0002) ("NT" 0.0)))
("tre-samhet" (("nn" 0.4425) ("pn" 0.3612) ("al" 0.0977) ("ab" 0.045) ("vb" 0.021) ("pm" 0.0187) ("av" 0.0121) ("**" 0.0017) ("an" 0.0001) ("NT" 0.0001)))
("två-samhet" (("nn" 0.4425) ("pn" 0.3612) ("al" 0.0977) ("ab" 0.045) ("vb" 0.021) ("pm" 0.0187) ("av" 0.0121) ("**" 0.0017) ("an" 0.0001) ("NT" 0.0001)))
("tvåsam-tresam-ensam" (("ab" 0.4135) ("av" 0.2023) ("nn" 0.197) ("pm" 0.1769) ("**" 0.0098) ("an" 0.0005)))
("upp-och-ner" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab" 0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018) ("NT" 0.0003)))
("uppvarvning" (("nn" 0.8346) ("pm" 0.0343) ("av" 0.0315) ("pp" 0.031) ("pn" 0.0296) ("ab" 0.0233) ("**" 0.0108) ("vb" 0.0039) ("NT" 0.0005) ("an" 0.0002) ("in" 0.0002)))
("urkass" (("pn" 0.6005) ("nn" 0.1549) ("av" 0.1208) ("ab" 0.0473) ("pm" 0.0453) ("**" 0.0166) ("vb" 0.0104) ("an" 0.0037) ("in" 0.0004)))
("varken-jag-eller-du" (("pn" 0.7436) ("**" 0.1987) ("pm" 0.0321) ("an" 0.0256)))
("visa-nivå" (("nl" 0.9198) ("nn" 0.0795) ("vb" 0.0008)))
("värstingar" (("vb" 0.6909) ("nn" 0.2646) ("pm" 0.0161) ("ab" 0.0113) ("av" 0.0108) ("pn" 0.0051) ("**" 0.0009) ("an" 0.0002) ("NT" 0.0)))

APPENDIX D – NFO AMENDMENTS AND ENTRIES LIFTED OUT PRIOR TO PROGRAM RUN

The test run exhibited some strange output. Pairs of *word class*/*frequency* like

```
1 (|ADJ" 8 "<>" "<adj>| 1)
2 ("religionsprocessen" 1)
3 ("pm0-s" 1)
4 ("nn1" 640)
5 (1 2)
```

... et cetera appeared, instead of desired and normal pairs, like for instance:

```
("nn" 312)
```

More instances than the above occured, but these six examples represent all types of undesirable output. In order to detect the source of the error, NFO was perused manually in its entirety. It was found that examples 1 and 2 could be attributed to corrupt format in NFO. A few compounds (not a great many, but a handful!) were so long that they required more than one line, and (for no apparent reason) the carriage return had resulted in incomplete entries. Rather than reconstructing these few entries in an *ad hoc* fashion, they were simply lifted out. The endings that these incomplete entries exhibited all exist in other words in the data base, so it is of minor importance for our present intents and purposes that NFO lacks a few entries.

Examples such as 3 and 4 were simply misprints, and all these were edited in a word processor (to enable full control over what was being carried out).

Example 5 occured when the NFO entry had no tag at all (not even the "**" foreign tag). Then the program – assuming a standard format – simply read the frequency as a word class. In order to solve this problem, a subroutine was added to make sure that the read 'word classes' actually *were* word classes. If the NFO entry did not fulfill this requirement (i.e., was not tagged) , the tag "NT", denoting 'Not Tagged (in NFO)', was provided by the program. The number of untagged entries in the computer version of NFO was not counted.

The following entries were lifted out since the words' final numbers are not *numbers* here, but NFO representations of diacritical signs. Since these were incompatible with SWETWOL format, it was decided to exclude the entries lest the percentage yield be inaccurate. ASCII format is not retained, but entries are given in Times 9 points to save space. NFO representation of |, *{*, *}* has been replaced by *ö*, *ä* and *å* respectively.

"blazkova1" 1 "<>" "<blazkova1>" "pm" "-s" 1
"pentekostae1" 1 "<>" "<pentekostae1>" "**" 1
"succe1" 13 "<>" "<succe1>" "nn" "-n" 21
"julboksflodsucce1" 1 "<>" "<julboksflodsucce1>" "nn" "-n" 1
"publiksucce1" 2 "<>" "<publiksucce1>" "nn" "-n" 4
"dundersucce1" 1 "<>" "<dundersucce1>" "nn" "-n" 2
"försäljningssucce1" 2 "<>" "<försäljningssucce1>" "nn" "-n" 2
"broadwaysucce1" 1 "<>" "<broadwaysucce1>" "nn" "-n" 1
"malmösucce1" 1 "<>" "<malmösucce1>" "nn" "-n" 1
"tierce1" 1 "<>" "<tierce1>" "**" 1
"amade1" 1 "<>" "<amade1>" "pm" "-s" 1
"ide1" 42 "<>" "<ide1>" "nn" "-n" 179
"bildide1" 1 "<>" "<bildide1>" "nn" "-n" 1
"rumside1" 1 "<>" "<rumside1>" "nn" "-n" 1
"helhetside1" 1 "<>" "<helhetside1>" "nn" "-n" 1
"cafe1" 5 "<>" "<cafe1>" "**" 5
"kafe1" 6 "<>" "<kafe1>" "nn" "-et" 17
"nattkafe1" 1 "<>" "<nattkafe1>" "nn" "-et" 1
"buffe1" 1 "<>" "<buffe1>" "nn" "-n" 2

"trofe1" 1 "<>" "<trofe1>" "nn" "-n" 3
"prote1ge1" 1 "<>" "<protege1>" "nn" "-n" 3
"drage1" 1 "<>" "<drage1>" "nn" "-n" 1
"protege1" 2 "<>" "<protege1>" "nn" "-n" 3
"kulturattache1" 1 "<>" "<kulturattache1>" "nn" "-n" 1
"militärattache1" 2 "<>" "<militärattache1>" "nn" "-n" 4
"kliche1" 2 "<>" "<kliche1>" "nn" "-n" 11
"saint-pierre-dumarche1" 1 "<>" "<saint-pierre-dumarche1>" "pm" "-s" 1
"lindhe1" 1 "<>" "<lindhe1>" "pm" "-s" 1
"conferencie1" 1 "<>" "<conferencie1>" "nn" "-n" 1
"foaje1" 3 "<>" "<foaje1>" "nn" "-n" 6
"hotellfoaje1" 2 "<>" "<hotellfoaje1>" "nn" "-n" 2
"atelje1" 7 "<>" "<atelje1>" "nn" "-n" 11
"kommunike1" 11 "<>" "<kommunike1>" "nn" "-n" 31
"bokslutskommunike1" 1 "<>" "<bokslutskommunike1>" "nn" "-n" 2
"slutkommunike1" 2 "<>" "<slutkommunike1>" "nn" "-n" 5
"gele1" 1 "<>" "<gele1>" "nn" "-n" 1
"file1" 3 "<>" "<file1>" "nn" "-n" 4
"koljafile1" 1 "<>" "<koljafile1>" "nn" "-n" 1
"ansjovisfile1" 1 "<>" "<ansjovisfile1>" "nn" "-n" 1
"spättfile1" 1 "<>" "<spättfile1>" "nn" "-n" 5
"oxfile1" 2 "<>" "<oxfile1>" "nn" "-n" 2
"galile1" 1 "<>" "<galile1>" "pm" "-s" 1
"aberle1" 1 "<>" "<aberle1>" "pm" "-s" 1
"final-lame1" 1 "<>" "<final-lame1>" "nn" "-n" 1
"silverlame1" 1 "<>" "<silverlame1>" "nn" "-n" 1
"aime1" 1 "<>" "<aime1>" "pm" "-s" 1
"renomme1" 2 "<>" "<renomme1>" "nn" "-et" 2
"arme1" 9 "<>" "<arme1>" "nn" "-n" 62
"befrielsearme1" 1 "<>" "<befrielsearme1>" "nn" "-n" 3
"kosackarme1" 1 "<>" "<kosackarme1>" "nn" "-n" 1
"mallarme1" 1 "<>" "<mallarme1>" "pm" "-s" 1
"partisanarme1" 2 "<>" "<partisanarme1>" "nn" "-n" 2
"miljonarme1" 1 "<>" "<miljonarme1>" "nn" "-n" 1
"turistarme1" 1 "<>" "<turistarme1>" "nn" "-n" 1
"resume1" 2 "<>" "<resume1>" "nn" "-n" 2
"rene1" 4 "<>" "<rene1>" "pm" "-s" 5
"matine1" 2 "<>" "<matine1>" "nn" "-n" 2
"lördagsmatine1" 1 "<>" "<lördagsmatine1>" "nn" "-n" 1
"linne1" 5 "<>" "<linne1>" "pm" "-s" 6
"turne1" 12 "<>" "<turne1>" "nn" "-n" 27
"propagandaturne1" 1 "<>" "<propagandaturne1>" "nn" "-n" 2
"amerikaturne1" 1 "<>" "<amerikaturne1>" "nn" "-n" 1
"talarturne1" 1 "<>" "<talarturne1>" "nn" "-n" 1
"sommarturne1" 1 "<>" "<sommarturne1>" "nn" "-n" 1
"utlandsturne1" 1 "<>" "<utlandsturne1>" "nn" "-n" 1
"mötesturne1" 1 "<>" "<mötesturne1>" "nn" "-n" 1
"konsertturne1" 1 "<>" "<konsertturne1>" "nn" "-n" 1
"rikskonsertturne1" 1 "<>" "<rikskonsertturne1>" "nn" "-n" 1
"yoritsune1" 1 "<>" "<yoritsune1>" "pm" "-s" 1
"procope1" 1 "<>" "<procope1>" "pm" "-s" 1
"europe1" 4 "<>" "<europe1>" "nn" "-n" 24
"kupe1" 2 "<>" "<kupe1>" "nn" "-n" 4
"förstaklasskupe1" 1 "<>" "<förstaklasskupe1>" "nn" "-n" 5
"supe1" 1 "<>" "<supe1>" "nn" "-n" 2
"krabb-supe1" 1 "<>" "<krabb-supe1>" "nn" "-n" 1
"teatersupe1" 1 "<>" "<teatersupe1>" "nn" "-n" 1
"moare1" 2 "<>" "<moare1>" "nn" "-n" 2
"debre1" 2 "<>" "<debre1>" "pm" "-s" 2
"andre1" 17 "<>" "<andre1>" "pm" "-s" 18
"carre1" 1 "<>" "<carre1>" "pm" "-s" 1
"ferre1" 2 "<>" "<ferre1>" "pm" "-s" 2
"entre1" 16 "<>" "<entre1>" "nn" "-n" 24
"världsentre1" 1 "<>" "<världsentre1>" "nn" "-n" 1
"cure1" 1 "<>" "<cure1>" "**" 1
"livre1" 2 "<>" "<livre1>" "nn" "-et" 2
"betjäntlivre1" 1 "<>" "<betjäntlivre1>" "nn" "-et" 1
"cayre1" 2 "<>" "<cayre1>" "pm" "-s" 2
"blase1" "obeutr" 1 "<>" "<blase1>" "av" "-=" 1

"blase1" 1 "<>" "<blase1>" "**" 1
"jose1" 5 "<>" "<jose1>" "pm" "-s" 5
"expose1" 4 "<>" "<expose1>" "nn" "-n" 5
"matelasse1" 2 "<>" "<matelasse1>" "nn" "-n" 2
"passe1" "obeutr" 1 "<>" "<passe1>" "av" "-=" 2
"passe1" "plu" 1 "<>" "<passe1>" "av" "-=" 2
"solfjädersplisse1" 1 "<>" "<solfjädersplisse1>" "nn" "-n" 1
"odysse1" 1 "<>" "<odysse1>" "pm" "-s" 1
"sekelskiftsvariete1" 1 "<>" "<sekelskiftsvariete1>" "nn" "-n" 1
"cite1" 1 "<>" "<cite1>" "**" 1
"specialite1" 1 "<>" "<specialite1>" "nn" "-n" 1
"kvalite1" 1 "<>" "<kvalite1>" "nn" "-n" 3
"tv-kvalite1" 1 "<>" "<tv-kvalite1>" "nn" "-n" 1
"l'humanite1" 2 "<>" "<l'humanite1>" "pm" "-s" 2
"clarte1" 1 "<>" "<clarte1>" "pm" "-s" 1
"kommitte1" 16 "<>" "<kommitte1>" "nn" "-n" 62
"kommunaltjänstemannakommitte1" 1 "<>" "<kommunaltjänstemannakommitte1>" "nn"
"högskolekommitte1" 1 "<>" "<högskolekommitte1>" "nn" "-n" 2
"sakkunnigkommitte1" 1 "<>" "<sakkunnigkommitte1>" "nn" "-n" 1
"affärsflygkommitte1" 1 "<>" "<affärsflygkommitte1>" "nn" "-n" 1
"specialkommitte1" 1 "<>" "<specialkommitte1>" "nn" "-n" 1
"centralkommitte1" 2 "<>" "<centralkommitte1>" "nn" "-n" 11
"ministerkommitte1" 1 "<>" "<ministerkommitte1>" "nn" "-n" 2
"urkommitte1" 1 "<>" "<urkommitte1>" "nn" "-n" 1
"utbildningskommitte1" 1 "<>" "<utbildningskommitte1>" "nn" "-n" 1
"förpackningskommitte1" 1 "<>" "<förpackningskommitte1>" "nn" "-n" 2
"planeringskommitte1" 1 "<>" "<planeringskommitte1>" "nn" "-n" 1
"samarbetskommitte1" 3 "<>" "<samarbetskommitte1>" "nn" "-n" 3
"kontaktkommitte1" 1 "<>" "<kontaktkommitte1>" "nn" "-n" 1
"expertkommitte1" 1 "<>" "<expertkommitte1>" "nn" "-n" 4
"frotte1" 1 "<>" "<frotte1>" "nn" "-n" 1
"sidendamastcloque1" 1 "<>" "<sidendamastcloque1>" "nn" "-n" 1
"sauve1" 1 "<>" "<sauve1>" "**" 1
"envoye1" 1 "<>" "<envoye1>" "nn" "-n" 2
"miro1" 7 "<>" "<miro1>" "pm" "-s" 8
"a2" 25 "<>" "<a2>" "**" 25
"a2" 38 "<>" "<a2>" "pp" 38
"l'unita2" 1 "<>" "<l'unita2>" "pm" "-s" 1
"verita2" 1 "<>" "<verita2>" "**" 1
"citta2" 1 "<>" "<citta2>" "**" 1
"calo2" 5 "<>" "<calo2>" "pm" "-s" 5
"vlno2" 1 "<>" "<vlno2>" "pm" "-s" 1
"e4" 1 "<>" "<e4>" "nn" "-n" 3
"chloe4" 2 "<>" "<chloe4>" "pm" "-s" 2

The following entries – which most likely emanate from misprints – were not tagged in NFO. Whereas other entries in NFO which are not tagged automatically are given the "NT" (i.e. 'Not Tagged') tag, it was felt that if only the figures *2*, *5* and *6* were given an additional tag (i.e., apart from the standard "an"), an inconsistent impression would be given. If, as is postulated, numbers behave in the same way (for good reasons, I dare say!), they should either *all* have an additional tag, or none should have an additional "NT" tag.

"juni<22" 1 "<>" "<juni<22>" 1
"mars<15" 1 "<>" "<mars<15>" 1
"juni<6" 1 "<>" "<juni<6>" 1

APPENDIX E – ENDING LEXICA

Here examples of the ending lexica employed by the module are given. In order to save space it is printed here in Times 9 points. The file employed by the module naturally has ASCII format. The lexica are also indented for the sake of legibility. Moreover, the two-letter to seven-letter lexica are columned to save space. The one-letter and two-letter lexica are given *in extenso*, whereas two page excerpts are given of ending lexica three to seven. Swedish letters *å*, *ä*, *ö* have been substituted for SWETWOL representation *}*, *{* and |.

ONE LETTER ENDING LEXICON

```
("'" (("pm" 0.7685) ("**" 0.1759) ("pn" 0.0185) ("vb" 0.0185) ("nn" 0.0093) ("an" 0.0093)))
(")" (("NT" 1.0)))
("-" (("NT" 1.0)))
("." (("an" 1.0)))
("0" (("an" 1.0)))
("1" (("an" 1.0)))
("2" (("an" 1.0)))
("3" (("an" 1.0)))
("4" (("an" 1.0)))
("5" (("an" 1.0)))
("6" (("an" 1.0)))
("7" (("an" 1.0)))
("8" (("an" 1.0)))
("9" (("an" 1.0)))
("a" (("av" 0.3179) ("vb" 0.2494) ("nn" 0.1618) ("pn" 0.1527) ("ab" 0.0522) ("pm" 0.0328) ("nl" 0.0157) ("an"
    0.0096) ("**" 0.004) ("in" 0.0024) ("pp" 0.0008) ("kn" 0.0005) ("NT" 0.0002) ("al" 0.0)))
("b" (("pm" 0.3406) ("nn" 0.2926) ("an" 0.2445) ("av" 0.0939) ("**" 0.0262) ("NT" 0.0022)))
("c" (("an" 0.6279) ("pm" 0.2869) ("**" 0.0607) ("nn" 0.023) ("NT" 0.0016)))
("d" (("pp" 0.4036) ("nn" 0.2392) ("vb" 0.1063) ("pm" 0.0762) ("ab" 0.0665) ("av" 0.0494) ("pn" 0.0453) ("**"
    0.0073) ("an" 0.0057) ("NT" 0.0003) ("in" 0.0001) ("kn" 0.0001)))
("e" (("vb" 0.3234) ("ab" 0.1802) ("nn" 0.1464) ("av" 0.1125) ("pn" 0.0765) ("al" 0.0688) ("pm" 0.0478) ("**"
    0.0182) ("nl" 0.0115) ("kn" 0.0098) ("pp" 0.0042) ("an" 0.0005) ("NT" 0.0002)))
("f" (("pm" 0.3584) ("nn" 0.3102) ("an" 0.1886) ("**" 0.1224) ("pp" 0.0171) ("in" 0.0016) ("av" 0.0008) ("vb"
    0.0008)))
("g" (("nn" 0.4509) ("pn" 0.2639) ("av" 0.1329) ("pm" 0.0483) ("ab" 0.0444) ("vb" 0.0373) ("pp" 0.0119) ("**"
    0.005) ("an" 0.0047) ("NT" 0.0005) ("in" 0.0002)))
("h" (("kn" 0.9759) ("pm" 0.0164) ("nn" 0.003) ("**" 0.003) ("an" 0.0011) ("in" 0.0003) ("av" 0.0002) ("NT"
    0.0001)))
("i" (("pp" 0.8059) ("pn" 0.082) ("nn" 0.0507) ("vb" 0.0269) ("pm" 0.018) ("**" 0.0041) ("av" 0.0037) ("ab"
    0.0034) ("nl" 0.0033) ("an" 0.0017) ("NT" 0.0002) ("in" 0.0001)))
("j" (("nn" 0.3053) ("ab" 0.2715) ("an" 0.1411) ("in" 0.1401) ("pm" 0.0928) ("pn" 0.0415) ("vb" 0.0029) ("NT"
    0.0029) ("av" 0.0019)))
("k" (("nn" 0.4526) ("av" 0.261) ("vb" 0.114) ("pm" 0.0817) ("ab" 0.0616) ("an" 0.0203) ("**" 0.0061) ("pn"
    0.0011) ("in" 0.0011) ("pp" 0.0004) ("NT" 0.0001)))
("l" (("pp" 0.351) ("nn" 0.306) ("vb" 0.1163) ("av" 0.0648) ("ab" 0.0618) ("pm" 0.0537) ("an" 0.0184) ("pn"
    0.0116) ("**" 0.0101) ("kn" 0.0057) ("in" 0.0002) ("NT" 0.0002) ("nl" 0.0001)))
("m" (("pn" 0.3602) ("kn" 0.2111) ("pp" 0.1864) ("nn" 0.0987) ("ab" 0.0544) ("pm" 0.0367) ("vb" 0.0204) ("av"
    0.0133) ("an" 0.0097) ("nl" 0.0059) ("**" 0.0029) ("NT" 0.0002) ("in" 0.0001)))
("n" (("nn" 0.3514) ("al" 0.1788) ("pn" 0.1765) ("kn" 0.0623) ("ab" 0.0604) ("pm" 0.0567) ("pp" 0.0445) ("vb"
    0.0341) ("nl" 0.0148) ("av" 0.0141) ("**" 0.005) ("an" 0.001) ("NT" 0.0002) ("in" 0.0001)))
("o" (("pm" 0.4039) ("nn" 0.2092) ("nl" 0.1162) ("an" 0.0642) ("vb" 0.0633) ("**" 0.0547) ("ab" 0.0538) ("in"
    0.0213) ("pn" 0.0067) ("av" 0.0061) ("NT" 0.0006)))
("p" (("nn" 0.5476) ("ab" 0.3308) ("pm" 0.0542) ("an" 0.026) ("av" 0.0143) ("vb" 0.0123) ("**" 0.0114) ("pn"
    0.0025) ("in" 0.0007) ("NT" 0.0002)))
("q" (("pm" 0.9) ("an" 0.1)))
("r" (("vb" 0.4392) ("nn" 0.2689) ("pp" 0.1379) ("ab" 0.0695) ("kn" 0.0355) ("pm" 0.0177) ("av" 0.015) ("an"
    0.0078) ("pn" 0.0051) ("**" 0.0034) ("NT" 0.0001) ("in" 0.0)))
("s" (("vb" 0.322) ("nn" 0.2815) ("pm" 0.1598) ("pn" 0.0899) ("ab" 0.0619) ("pp" 0.0264) ("an" 0.0167) ("**"
    0.0165) ("av" 0.0156) ("kn" 0.0088) ("nl" 0.0006) ("NT" 0.0003) ("in" 0.0001)))
("t" (("nn" 0.2081) ("ab" 0.1579) ("pn" 0.1429) ("vb" 0.1072) ("ie" 0.0868) ("al" 0.0824) ("av" 0.0815) ("kn"
    0.0812) ("pp" 0.024) ("pm" 0.0157) ("nl" 0.0043) ("an" 0.004) ("**" 0.0031) ("NT" 0.0007) ("in" 0.0001)))
("u" (("ab" 0.7944) ("nn" 0.0701) ("pm" 0.047) ("nl" 0.0257) ("**" 0.0257) ("pn" 0.025) ("kn" 0.0058) ("an"
    0.0054) ("NT" 0.0009) ("in" 0.0002)))
```

("v" (("nn" 0.3523) ("vb" 0.2551) ("av" 0.2219) ("ab" 0.0673) ("pm" 0.0414) ("an" 0.0321) ("nl" 0.0143) ("pn" 0.0142) ("NT" 0.0011) ("**" 0.0002) ("pp" 0.0002)))
("w" (("**" 0.6774) ("pm" 0.2053) ("an" 0.085) ("nn" 0.0293) ("NT" 0.0029)))
("x" (("an" 0.4159) ("nl" 0.1866) ("pm" 0.1394) ("nn" 0.1233) ("ab" 0.0956) ("**" 0.0265) ("av" 0.0069) ("NT" 0.0058)))
("y" (("pm" 0.4595) ("av" 0.2071) ("**" 0.1433) ("kn" 0.0919) ("nn" 0.0752) ("vb" 0.0124) ("an" 0.0057) ("ab" 0.0019) ("pn" 0.0014) ("in" 0.001) ("NT" 0.0005)))
("z" (("pm" 0.8687) ("nn" 0.0687) ("**" 0.0507) ("an" 0.0119)))
("ä" (("nn" 0.6311) ("vb" 0.123) ("pm" 0.0902) ("an" 0.0738) ("in" 0.0656) ("**" 0.0164)))
("ö" (("pm" 0.4809) ("nn" 0.3876) ("an" 0.0806) ("vb" 0.0325) ("in" 0.0113) ("NT" 0.0042) ("av" 0.0014) ("pn" 0.0014)))
("å" (("pp" 0.4846) ("ab" 0.3178) ("vb" 0.092) ("nl" 0.041) ("kn" 0.0296) ("av" 0.0202) ("nn" 0.0067) ("pn" 0.0043) ("pm" 0.0022) ("in" 0.0015) ("**" 0.0001) ("an" 0.0001)))

TWO LETTER ENDING LEXICON

```
("a'" (("vb" 0.6667) ("**" 0.3333)))
("e'" (("pn" 1.0)))
("h'" (("pm" 1.0)))
("n'" (("**" 1.0)))
("r'" (("**" 1.0)))
("s'" (("pm" 0.8642) ("**" 0.1111) ("nn" 0.0123)
    ("an" 0.0123)))
("x'" (("pm" 1.0)))
("z'" (("pm" 1.0)))
("t)" (("NT" 1.0)))
("1-" (("NT" 1.0)))
("a-" (("NT" 1.0)))
("b-" (("NT" 1.0)))
("d-" (("NT" 1.0)))
("e-" (("NT" 1.0)))
("f-" (("NT" 1.0)))
("g-" (("NT" 1.0)))
("i-" (("NT" 1.0)))
("k-" (("NT" 1.0)))
("l-" (("NT" 1.0)))
("m-" (("NT" 1.0)))
("n-" (("NT" 1.0)))
("o-" (("NT" 1.0)))
("p-" (("NT" 1.0)))
("r-" (("NT" 1.0)))
("s-" (("NT" 1.0)))
("t-" (("NT" 1.0)))
("u-" (("NT" 1.0)))
("v-" (("NT" 1.0)))
("y-" (("NT" 1.0)))
("ö-" (("NT" 1.0)))
("å-" (("NT" 1.0)))
("a." (("an" 1.0)))
("b." (("an" 1.0)))
("c." (("an" 1.0)))
("d." (("an" 1.0)))
("e." (("an" 1.0)))
("f." (("an" 1.0)))
("g." (("an" 1.0)))
("h." (("an" 1.0)))
("j." (("an" 1.0)))
("k." (("an" 1.0)))
("l." (("an" 1.0)))
("m." (("an" 1.0)))
("n." (("an" 1.0)))
("o." (("an" 1.0)))
("p." (("an" 1.0)))
("r." (("an" 1.0)))
("s." (("an" 1.0)))
("t." (("an" 1.0)))
("u." (("an" 1.0)))
("v." (("an" 1.0)))
("w." (("an" 1.0)))
("x." (("an" 1.0)))
("y." (("an" 1.0)))
("z." (("an" 1.0)))
("ä." (("an" 1.0)))
("ö." (("an" 1.0)))
("10" (("an" 1.0)))
("30" (("an" 1.0)))
("11" (("an" 1.0)))
("21" (("an" 1.0)))
("31" (("an" 1.0)))
("41" (("an" 1.0)))
("51" (("an" 1.0)))
("81" (("an" 1.0)))
("-2" (("an" 1.0)))
("62" (("an" 1.0)))
("f2" (("an" 1.0)))
("m2" (("an" 1.0)))
("p2" (("an" 1.0)))
("53" (("an" 1.0)))
("63" (("an" 1.0)))
("n3" (("an" 1.0)))
("p3" (("an" 1.0)))
("y3" (("an" 1.0)))
("34" (("an" 1.0)))
("44" (("an" 1.0)))
("54" (("an" 1.0)))
("15" (("an" 1.0)))
("/6" (("an" 1.0)))
("16" (("an" 1.0)))
("18" (("an" 1.0)))
("-9" (("an" 1.0)))
("1a" (("pm" 0.6667) ("**" 0.3333)))
("5a" (("pm" 1.0)))
(":a" (("an" 1.0)))
("aa" (("**" 0.6) ("pm" 0.4)))
("ba" (("pm" 0.4233) ("av" 0.3333) ("vb" 0.1746)
    ("nn" 0.037) ("**" 0.0159) ("an" 0.0159)))
("ca" (("an" 0.5952) ("pm" 0.3045) ("**" 0.0969)
    ("nn" 0.0035)))
("da" (("vb" 0.3831) ("av" 0.3634) ("nn" 0.0901)
    ("pn" 0.0674) ("ab" 0.0629) ("pm" 0.0231)
    ("kn" 0.0067) ("**" 0.0024) ("an" 0.0004)
    ("NT" 0.0003) ("in" 0.0001)))
("ea" (("pm" 0.7727) ("**" 0.1136) ("nn" 0.0795)
    ("an" 0.0341)))
("fa" (("vb" 0.8814) ("nn" 0.0515) ("pm" 0.0515)
    ("av" 0.0155)))
("ga" (("av" 0.5724) ("pn" 0.1803) ("vb" 0.1468)
    ("nn" 0.0778) ("ab" 0.0155) ("pm" 0.007) ("**"
    0.0001) ("NT" 0.0001) ("an" 0.0001)))
("ha" (("vb" 0.9719) ("pm" 0.0124) ("in" 0.0072)
    ("an" 0.0046) ("av" 0.002) ("**" 0.0013) ("nn"
    0.0007)))
("ia" (("pm" 0.3945) ("nn" 0.3165) ("av" 0.1625)
    ("pp" 0.0817) ("**" 0.0219) ("vb" 0.019) ("an"
    0.0038)))
("ja" (("vb" 0.6203) ("in" 0.1757) ("nn" 0.1496)
    ("pm" 0.051) ("**" 0.0034)))
("ka" (("av" 0.6322) ("vb" 0.1411) ("ab" 0.0956)
    ("nn" 0.074) ("pn" 0.0344) ("pm" 0.0227)))
("la" (("av" 0.4597) ("vb" 0.2149) ("pn" 0.1864)
    ("nn" 0.0726) ("pm" 0.0389) ("ab" 0.0141)
    ("**" 0.0129) ("NT" 0.0003) ("in" 0.0001)))
("ma" (("vb" 0.3178) ("pn" 0.2944) ("av" 0.1912)
    ("nn" 0.0975) ("ab" 0.0584) ("pm" 0.035) ("**"
    0.0053) ("in" 0.0003)))
("na" (("nn" 0.5513) ("pn" 0.1789) ("vb" 0.1359)
    ("av" 0.0897) ("pm" 0.0312) ("ab" 0.0117)
    ("**" 0.0008) ("an" 0.0003) ("NT" 0.0002) ("al"
    0.0)))
("oa" (("vb" 0.5714) ("pm" 0.381) ("**" 0.0476)))
("pa" (("vb" 0.6322) ("pm" 0.1806) ("nn" 0.1371)
    ("av" 0.0394) ("ab" 0.0049) ("**" 0.0033) ("pn"
    0.0025)))
("ra" (("vb" 0.3749) ("pn" 0.2303) ("av" 0.1758)
    ("ab" 0.1439) ("nl" 0.0407) ("nn" 0.0207) ("pm"
    0.0096) ("**" 0.0022) ("kn" 0.0009) ("pp"
    0.0005) ("NT" 0.0002) ("in" 0.0001) ("an"
    0.0001)))
```

("sa" (("pn" 0.2977) ("vb" 0.2439) ("av" 0.2401) ("nn" 0.0977) ("an" 0.0955) ("pm" 0.0216) ("**" 0.0033) ("NT" 0.0003)))
("ta" (("av" 0.3081) ("vb" 0.2998) ("pn" 0.1718) ("nl" 0.0788) ("nn" 0.0528) ("ab" 0.0523) ("pm" 0.0301) ("**" 0.0042) ("an" 0.002) ("NT" 0.0002)))
("ua" (("pm" 0.4737) ("**" 0.3947) ("vb" 0.1053) ("nn" 0.0263)))
("va" (("av" 0.51) ("vb" 0.3227) ("pm" 0.0801) ("nn" 0.0522) ("pn" 0.0155) ("nl" 0.0128) ("**" 0.0062) ("NT" 0.0004)))
("wa" (("pm" 1.0)))
("xa" (("vb" 0.6364) ("av" 0.1818) ("nn" 0.1667) ("pm" 0.0152)))
("ya" (("av" 0.9494) ("pm" 0.0307) ("vb" 0.0124) ("nn" 0.005) ("**" 0.0025)))
("za" (("pm" 0.5294) ("nn" 0.2647) ("**" 0.2059)))
("öa" (("an" 0.6667) ("vb" 0.3333)))
("åa" (("av" 0.7917) ("nn" 0.2083)))
("ab" (("an" 0.5354) ("pm" 0.3636) ("nn" 0.0808) ("**" 0.0101) ("NT" 0.0101)))
("bb" (("nn" 0.6515) ("av" 0.3258) ("an" 0.0152) ("pm" 0.0076)))
("db" (("an" 1.0)))
("eb" (("pm" 0.6) ("**" 0.4)))
("gb" (("an" 1.0)))
("hb" (("an" 1.0)))
("ib" (("pm" 0.6364) ("an" 0.2727) ("**" 0.0909)))
("kb" (("an" 1.0)))
("lb" (("pm" 1.0)))
("mb" (("nn" 0.9091) ("pm" 0.0909)))
("nb" (("an" 1.0)))
("ob" (("pm" 0.8478) ("nn" 0.1087) ("**" 0.0435)))
("rb" (("nn" 0.5) ("an" 0.375) ("pm" 0.125)))
("sb" (("an" 1.0)))
("ub" (("pm" 0.5806) ("**" 0.1935) ("nn" 0.1935) ("an" 0.0323)))
("öb" (("an" 1.0)))
("åb" (("nn" 1.0)))
("1c" (("pm" 1.0)))
("ac" (("pm" 0.9737) ("**" 0.0263)))
("bc" (("an" 1.0)))
("cc" (("an" 1.0)))
("dc" (("an" 1.0)))
("ec" (("an" 0.7215) ("pm" 0.2785)))
("fc" (("an" 1.0)))
("ic" (("pm" 0.6744) ("**" 0.2558) ("an" 0.062) ("NT" 0.0078)))
("mc" (("an" 1.0)))
("nc" (("nn" 0.6087) ("pm" 0.2609) ("**" 0.087) ("an" 0.0435)))
("oc" (("an" 0.9871) ("pm" 0.0129)))
("rc" (("pm" 0.75) ("an" 0.25)))
("tc" (("an" 1.0)))
("uc" (("pm" 0.8) ("**" 0.2)))
("wc" (("an" 1.0)))
("'d" (("**" 1.0)))
("-d" (("NT" 1.0)))
("ad" (("vb" 0.3732) ("pn" 0.2646) ("nn" 0.233) ("av" 0.1022) ("pm" 0.0253) ("**" 0.0009) ("NT" 0.0005) ("ab" 0.0003)))
("bd" (("an" 1.0)))
("cd" (("an" 1.0)))
("dd" (("vb" 0.6248) ("nn" 0.2345) ("av" 0.1295) ("pm" 0.0113)))
("ed" (("pp" 0.902) ("ab" 0.0609) ("nn" 0.0179) ("pm" 0.0078) ("vb" 0.0043) ("av" 0.0035) ("**" 0.0029) ("an" 0.0006)))
("gd" (("vb" 0.4521) ("nn" 0.4438) ("av" 0.1014) ("NT" 0.0027)))
("hd" (("an" 1.0)))
("id" (("pp" 0.5011) ("ab" 0.2359) ("nn" 0.2313) ("pm" 0.0234) ("av" 0.0059) ("**" 0.0016) ("an" 0.0006) ("vb" 0.0002)))
("jd" (("nn" 0.7532) ("av" 0.1266) ("vb" 0.1171) ("NT" 0.0032)))
("ld" (("nn" 0.5533) ("vb" 0.2107) ("pm" 0.1456) ("av" 0.0724) ("**" 0.0179)))
("md" (("vb" 0.5515) ("av" 0.3088) ("nn" 0.1397)))
("nd" (("nn" 0.4274) ("pm" 0.3019) ("pp" 0.1205) ("ab" 0.0703) ("av" 0.0354) ("**" 0.0282) ("vb" 0.0131) ("an" 0.0018) ("NT" 0.0009) ("kn" 0.0004)))
("od" (("vb" 0.3688) ("nn" 0.2857) ("av" 0.2735) ("pm" 0.0576) ("**" 0.0122) ("in" 0.0022)))
("pd" (("an" 1.0)))
("rd" (("nn" 0.5306) ("pm" 0.2029) ("av" 0.1408) ("vb" 0.0782) ("ab" 0.0337) ("**" 0.0128) ("an" 0.0005) ("NT" 0.0005)))
("sd" (("an" 1.0)))
("td" (("an" 1.0)))
("ud" (("nn" 0.6971) ("pm" 0.2183) ("ab" 0.0423) ("an" 0.0334) ("**" 0.0045) ("vb" 0.0022) ("in" 0.0022)))
("vd" (("an" 0.7887) ("vb" 0.1408) ("av" 0.0423) ("nn" 0.0282)))
("yd" (("nn" 0.7273) ("pm" 0.2597) ("vb" 0.013)))
("äd" (("nn" 0.9706) ("av" 0.0294)))
("öd" (("nn" 0.7257) ("av" 0.1586) ("vb" 0.1082) ("pm" 0.0075)))
("åd" (("nn" 0.9907) ("**" 0.0046) ("av" 0.0046)))
("1e" (("pm" 0.5882) ("**" 0.4118)))
(":e" (("nl" 0.5) ("an" 0.5)))
("ae" (("**" 0.5385) ("pm" 0.3846) ("an" 0.0769)))
("be" (("pm" 0.6491) ("vb" 0.1579) ("nn" 0.0965) ("**" 0.0965)))
("ce" (("pm" 0.4357) ("**" 0.2714) ("nn" 0.1679) ("pn" 0.125)))
("de" (("vb" 0.5279) ("al" 0.1503) ("pn" 0.1319) ("nn" 0.0735) ("av" 0.0682) ("kn" 0.0184) ("ab" 0.0163) ("**" 0.0085) ("pm" 0.0025) ("nl" 0.0023) ("NT" 0.0001) ("an" 0.0001)))
("ee" (("pm" 0.8404) ("**" 0.1596)))
("fe" (("nn" 0.75) ("pm" 0.1912) ("**" 0.0588)))
("ge" (("pm" 0.352) ("vb" 0.2124) ("nn" 0.1658) ("ab" 0.1589) ("av" 0.095) ("**" 0.0152) ("NT" 0.0007)))
("he" (("**" 0.8052) ("pm" 0.1891) ("nn" 0.0057)))
("ie" (("nn" 0.4141) ("pm" 0.3616) ("**" 0.1596) ("av" 0.0566) ("pn" 0.0081)))
("je" (("pn" 0.6006) ("nn" 0.2362) ("nl" 0.1307) ("pm" 0.0317) ("pp" 0.0008)))
("ke" (("ab" 0.4687) ("nn" 0.1829) ("av" 0.1555) ("pm" 0.1548) ("vb" 0.0324) ("**" 0.005) ("NT" 0.0007)))
("le" (("vb" 0.6884) ("pm" 0.1242) ("nn" 0.116) ("**" 0.0401) ("av" 0.0296) ("an" 0.0008) ("pn" 0.0006) ("ab" 0.0003)))
("me" (("nn" 0.4961) ("**" 0.213) ("pm" 0.1273) ("ab" 0.0701) ("av" 0.0597) ("pn" 0.0208) ("vb" 0.0078) ("an" 0.0052)))
("ne" (("pm" 0.284) ("pn" 0.2316) ("ab" 0.206) ("nn" 0.1622) ("av" 0.0472) ("**" 0.0438) ("vb" 0.0205) ("nl" 0.0034) ("NT" 0.0011)))
("oe" (("pm" 0.8333) ("nn" 0.1667)))
("pe" (("ab" 0.5591) ("pm" 0.2151) ("nn" 0.1452) ("**" 0.0753) ("av" 0.0054)))

```
("re" (("av" 0.3383) ("nn" 0.3065) ("ab" 0.2201)
    ("nl" 0.0516) ("vb" 0.0271) ("pp" 0.0229) ("pm"
    0.02) ("**" 0.0064) ("kn" 0.0051) ("pn" 0.0016)
    ("NT" 0.0005)))
("se" (("nn" 0.6465) ("vb" 0.2461) ("pm" 0.0534)
    ("av" 0.0218) ("pn" 0.0152) ("**" 0.0152) ("ab"
    0.0015) ("NT" 0.0004)))
("te" (("ab" 0.6264) ("vb" 0.1703) ("av" 0.096)
    ("nn" 0.0778) ("pm" 0.0139) ("**" 0.0062)
    ("pn" 0.0039) ("nl" 0.0033) ("pp" 0.0021) ("NT"
    0.0001)))
("ue" (("**" 0.5392) ("pm" 0.4118) ("nn" 0.049)))
("ve" (("pm" 0.3498) ("kn" 0.1486) ("**" 0.1455)
    ("nn" 0.1084) ("pp" 0.0836) ("av" 0.0619) ("vb"
    0.0526) ("pn" 0.0495)))
("we" (("pm" 0.6957) ("**" 0.3043)))
("xe" (("**" 0.5) ("nn" 0.5)))
("ye" (("av" 0.7917) ("**" 0.1042) ("pm" 0.1042)))
("ze" (("pm" 0.5556) ("**" 0.2222) ("nn" 0.2222)))
("öe" (("nn" 0.6667) ("av" 0.3333)))
("af" (("pm" 0.4718) ("nn" 0.3521) ("pp" 0.1479)
    ("an" 0.0141) ("**" 0.007) ("vb" 0.007)))
("bf" (("an" 1.0)))
("df" (("an" 1.0)))
("ef" (("nn" 0.8832) ("pm" 0.0914) ("**" 0.0254)))
("ff" (("nn" 0.6552) ("pm" 0.3034) ("**" 0.0207)
    ("in" 0.0138) ("av" 0.0069)))
("hf" (("an" 1.0)))
("if" (("pm" 0.9032) ("**" 0.0968)))
("jf" (("nn" 1.0)))
("kf" (("an" 1.0)))
("lf" (("pm" 0.875) ("an" 0.0903) ("**" 0.0208)
    ("nn" 0.0139)))
("mf" (("nn" 0.9) ("an" 0.1)))
("nf" (("an" 1.0)))
("of" (("**" 0.5216) ("pm" 0.3294) ("nn" 0.1255)
    ("an" 0.0235)))
("pf" (("pm" 0.8333) ("**" 0.1667)))
("rf" (("pm" 0.4194) ("an" 0.3226) ("nn" 0.2581)))
("sf" (("an" 1.0)))
("tf" (("an" 1.0)))
("uf" (("an" 0.7778) ("**" 0.1111) ("pm" 0.1111)))
("öf" (("pm" 1.0)))
("-g" (("an" 1.0)))
("ag" (("nn" 0.5004) ("pn" 0.4728) ("ab" 0.0088)
    ("av" 0.0066) ("pm" 0.0063) ("vb" 0.0024)
    ("**" 0.0013) ("NT" 0.0006) ("in" 0.0006) ("an"
    0.0004)))
("dg" (("an" 1.0)))
("eg" (("nn" 0.6733) ("vb" 0.2327) ("pm" 0.0594)
    ("**" 0.0099) ("av" 0.0099) ("an" 0.0099)
    ("NT" 0.005)))
("gg" (("nn" 0.8127) ("av" 0.0919) ("vb" 0.0777)
    ("pm" 0.0177)))
("hg" (("an" 0.6667) ("pm" 0.3333)))
("ig" (("pn" 0.5671) ("av" 0.3503) ("ab" 0.0542)
    ("nn" 0.0145) ("pm" 0.0128) ("**" 0.0006)
    ("NT" 0.0003) ("vb" 0.0002) ("an" 0.0001)))
("kg" (("an" 1.0)))
("lg" (("nn" 0.9) ("**" 0.1)))
("mg" (("an" 1.0)))
("ng" (("nn" 0.8346) ("pm" 0.0343) ("av" 0.0315)
    ("pp" 0.031) ("pn" 0.0296) ("ab" 0.0233) ("**"
    0.0108) ("vb" 0.0039) ("NT" 0.0005) ("an"
    0.0002) ("in" 0.0002)))
("og" (("vb" 0.4667) ("ab" 0.4045) ("nn" 0.1136)
    ("**" 0.0091) ("pm" 0.0061)))
("pg" (("an" 1.0)))
("rg" (("pm" 0.7976) ("nn" 0.192) ("av" 0.0088)
    ("NT" 0.0016)))
("ug" (("nn" 0.55) ("av" 0.25) ("an" 0.1) ("**" 0.05)
    ("pm" 0.05)))
("yg" (("nn" 0.9415) ("av" 0.039) ("pm" 0.0139)
    ("**" 0.0028) ("vb" 0.0028)))
("äg" (("nn" 0.9) ("ab" 0.075) ("vb" 0.025)))
("ög" (("av" 0.8421) ("vb" 0.1158) ("nn" 0.0246)
    ("pm" 0.0175)))
("åg" (("vb" 0.7481) ("nn" 0.1551) ("av" 0.0614)
    ("ab" 0.0353)))
("/h" (("an" 1.0)))
("ah" (("pm" 0.7857) ("**" 0.1429) ("in" 0.0714)))
("bh" (("**" 1.0)))
("ch" (("kn" 0.9907) ("pm" 0.0049) ("nn" 0.0029)
    ("**" 0.0011) ("av" 0.0002) ("NT" 0.0001) ("in"
    0.0)))
("dh" (("pm" 1.0)))
("eh" (("**" 0.5) ("pm" 0.5)))
("gh" (("pm" 0.6552) ("**" 0.3103) ("nn" 0.0345)))
("hh" (("an" 1.0)))
("kh" (("pm" 1.0)))
("lh" (("pm" 1.0)))
("nh" (("pm" 1.0)))
("oh" (("in" 0.625) ("**" 0.375)))
("ph" (("pm" 0.9615) ("**" 0.0385)))
("rh" (("an" 1.0)))
("sh" (("**" 0.5952) ("pm" 0.2857) ("nn" 0.0952)
    ("an" 0.0238)))
("th" (("pm" 0.9246) ("**" 0.0714) ("an" 0.004)))
("uh" (("an" 0.5) ("pm" 0.5)))
("'i" (("**" 1.0)))
("8i" (("pm" 1.0)))
("ai" (("pm" 0.8571) ("**" 0.1429)))
("bi" (("ab" 0.4959) ("pp" 0.2276) ("an" 0.1789)
    ("pm" 0.0732) ("nn" 0.0244)))
("ci" (("pm" 0.6667) ("**" 0.3333)))
("di" (("nn" 0.6028) ("pm" 0.2482) ("**" 0.1348)
    ("pn" 0.0142)))
("ei" (("pm" 0.7447) ("**" 0.1489) ("an" 0.1064)))
("fi" (("nn" 0.9272) ("pm" 0.0596) ("an" 0.0132)))
("gi" (("nn" 0.9231) ("pm" 0.0737) ("**" 0.0032)))
("hi" (("pm" 0.931) ("nn" 0.0345) ("**" 0.0345)))
("ii" (("nl" 0.9174) ("pm" 0.0459) ("**" 0.0275)
    ("NT" 0.0092)))
("ji" (("pm" 0.6667) ("pn" 0.3333)))
("ki" (("pm" 0.8462) ("nn" 0.0962) ("**" 0.0577)))
("li" (("vb" 0.8707) ("nn" 0.0638) ("pm" 0.0576)
    ("**" 0.0062) ("NT" 0.0018)))
("mi" (("nn" 0.8189) ("pm" 0.1811)))
("ni" (("pn" 0.4879) ("nn" 0.3006) ("pm" 0.1967)
    ("**" 0.013) ("NT" 0.0019)))
("oi" (("pm" 0.8) ("**" 0.2)))
("pi" (("nn" 0.6765) ("pm" 0.2353) ("an" 0.0882)))
("ri" (("nn" 0.6886) ("av" 0.1546) ("pm" 0.0784)
    ("ab" 0.0611) ("**" 0.0127) ("an" 0.0046)))
("si" (("nn" 0.6591) ("pm" 0.1818) ("**" 0.0758)
    ("in" 0.0379) ("ab" 0.0379) ("NT" 0.0076)))
("ti" (("nn" 0.7459) ("pm" 0.1878) ("pp" 0.0166)
    ("ab" 0.0138) ("**" 0.0138) ("pn" 0.011) ("NT"
    0.0055) ("nl" 0.0055)))
("ui" (("nn" 0.4286) ("pm" 0.2857) ("**" 0.2857)))
("vi" (("pn" 0.9884) ("pm" 0.0072) ("nl" 0.0018)
    ("pp" 0.0014) ("**" 0.0011)))
("wi" (("pm" 1.0)))
("xi" (("nn" 0.85) ("nl" 0.1) ("**" 0.05)))
("yi" (("pm" 1.0)))
("zi" (("pm" 0.75) ("**" 0.25)))
("1j" (("pm" 1.0)))
```

("aj" (("nn" 0.8273) ("pm" 0.1364) ("NT" 0.0273) ("in" 0.0091)))
("dj" (("pm" 1.0)))
("ej" (("ab" 0.5553) ("in" 0.2826) ("pn" 0.085) ("nn" 0.0494) ("pm" 0.0277)))
("ij" (("pm" 0.9787) ("av" 0.0213)))
("lj" (("nn" 0.6109) ("an" 0.3818) ("vb" 0.0073)))
("nj" (("nn" 1.0)))
("oj" (("pm" 0.3636) ("nn" 0.3636) ("an" 0.1818) ("av" 0.0455) ("in" 0.0455)))
("sj" (("an" 1.0)))
("tj" (("pm" 1.0)))
("yj" (("pm" 1.0)))
("öj" (("pm" 0.5) ("vb" 0.5)))
("1k" (("pm" 1.0)))
("7k" (("pm" 1.0)))
("8k" (("**" 0.5) ("pm" 0.5)))
("ak" (("nn" 0.8612) ("pm" 0.0607) ("av" 0.026) ("an" 0.0217) ("ab" 0.0195) ("pp" 0.0108)))
("ck" (("vb" 0.3963) ("nn" 0.2872) ("ab" 0.2346) ("pm" 0.0603) ("av" 0.0088) ("**" 0.007) ("in" 0.0041) ("an" 0.0018)))
("ek" (("nn" 0.7618) ("pm" 0.116) ("vb" 0.0752) ("av" 0.0408) ("**" 0.0063)))
("fk" (("an" 1.0)))
("hk" (("an" 1.0)))
("ik" (("nn" 0.7484) ("pm" 0.148) ("av" 0.0905) ("pn" 0.0078) ("**" 0.0047) ("NT" 0.0005)))
("jk" (("pm" 0.5938) ("nn" 0.4062)))
("kk" (("pm" 0.7273) ("an" 0.2727)))
("lk" (("nn" 0.97) ("pm" 0.0253) ("an" 0.0023) ("**" 0.0023)))
("mk" (("an" 1.0)))
("nk" (("nn" 0.4292) ("vb" 0.2689) ("pm" 0.1792) ("**" 0.0519) ("an" 0.0425) ("av" 0.0283)))
("ok" (("nn" 0.8901) ("pm" 0.0581) ("av" 0.0298) ("**" 0.0173) ("an" 0.0047)))
("pk" (("an" 1.0)))
("rk" (("nn" 0.5726) ("pm" 0.278) ("av" 0.1311) ("**" 0.0125) ("an" 0.0033) ("vb" 0.0025)))
("sk" (("av" 0.922) ("nn" 0.0655) ("pm" 0.0059) ("an" 0.004) ("**" 0.0022) ("ab" 0.0003)))
("tk" (("an" 1.0)))
("uk" (("nn" 0.7452) ("av" 0.2357) ("pm" 0.019)))
("wk" (("pm" 1.0)))
("yk" (("pm" 0.5455) ("nn" 0.4545)))
("äk" (("nn" 1.0)))
("ök" (("nn" 0.8585) ("vb" 0.1185) ("pm" 0.0229)))
("åk" (("nn" 0.9963) ("vb" 0.0037)))
("4l" (("pm" 1.0)))
("al" (("nn" 0.6618) ("av" 0.2237) ("pm" 0.062) ("**" 0.0471) ("vb" 0.0042) ("an" 0.0011)))
("bl" (("an" 1.0)))
("dl" (("an" 0.95) ("pm" 0.05)))
("el" (("nn" 0.7582) ("pm" 0.1134) ("av" 0.0905) ("ab" 0.0176) ("**" 0.0163) ("pn" 0.0028) ("an" 0.001) ("NT" 0.0003)))
("fl" (("an" 1.0)))
("gl" (("an" 0.6667) ("pm" 0.3333)))
("hl" (("pm" 0.9778) ("an" 0.0222)))
("il" (("nn" 0.7465) ("pm" 0.1282) ("av" 0.0535) ("**" 0.038) ("an" 0.031) ("NT" 0.0028)))
("kl" (("an" 1.0)))
("ll" (("pp" 0.5629) ("vb" 0.1839) ("nn" 0.1253) ("av" 0.0454) ("ab" 0.0376) ("pm" 0.023) ("pn" 0.018) ("**" 0.0021) ("kn" 0.0014) ("nl" 0.0002) ("an" 0.0001) ("NT" 0.0001) ("in" 0.0001)))
("ml" (("pm" 0.9286) ("an" 0.0714)))
("nl" (("an" 1.0)))
("ol" (("nn" 0.8264) ("pm" 0.0903) ("**" 0.0694) ("an" 0.0104) ("vb" 0.0035)))
("rl" (("pm" 0.813) ("nn" 0.1609) ("**" 0.0217) ("an" 0.0043)))
("sl" (("an" 1.0)))
("ul" (("pm" 0.4468) ("nn" 0.3191) ("av" 0.1915) ("**" 0.0372) ("NT" 0.0053)))
("wl" (("**" 1.0)))
("xl" (("an" 1.0)))
("yl" (("nn" 0.8) ("an" 0.1429) ("pm" 0.0571)))
("äl" (("ab" 0.7093) ("nn" 0.1782) ("kn" 0.095) ("av" 0.0105) ("in" 0.0042) ("vb" 0.0028)))
("öl" (("nn" 0.9826) ("**" 0.0087) ("pm" 0.0087)))
("ål" (("nn" 0.9204) ("vb" 0.0583) ("pm" 0.0136) ("av" 0.0058) ("**" 0.0019)))
("'m" (("**" 1.0)))
("3m" (("nn" 1.0)))
("am" (("ab" 0.4135) ("av" 0.2023) ("nn" 0.197) ("pm" 0.1769) ("**" 0.0098) ("an" 0.0005)))
("bm" (("an" 1.0)))
("cm" (("an" 1.0)))
("dm" (("an" 1.0)))
("em" (("pn" 0.5015) ("nn" 0.3256) ("nl" 0.0975) ("ab" 0.0537) ("pm" 0.0145) ("av" 0.0038) ("**" 0.0023) ("NT" 0.0011)))
("gm" (("nn" 1.0)))
("hm" (("pm" 0.6) ("**" 0.35) ("in" 0.05)))
("im" (("pm" 0.4215) ("nn" 0.3058) ("an" 0.1322) ("av" 0.0909) ("**" 0.0413) ("in" 0.0083)))
("jm" (("an" 1.0)))
("km" (("an" 1.0)))
("lm" (("pm" 0.7153) ("nn" 0.2614) ("an" 0.0163) ("NT" 0.0035) ("**" 0.0035)))
("mm" (("an" 0.4737) ("nn" 0.2807) ("pm" 0.2105) ("**" 0.0351)))
("om" (("pn" 0.4161) ("kn" 0.2662) ("pp" 0.235) ("ab" 0.0401) ("vb" 0.0254) ("nn" 0.0115) ("pm" 0.0038) ("av" 0.001) ("**" 0.0008) ("NT" 0.0001) ("in" 0.0)))
("pm" (("an" 1.0)))
("rm" (("nn" 0.8985) ("av" 0.0667) ("pm" 0.0273) ("**" 0.0076)))
("sm" (("nn" 0.9955) ("an" 0.0045)))
("tm" (("nn" 1.0)))
("um" (("nn" 0.8861) ("pm" 0.0608) ("**" 0.0374) ("av" 0.014) ("NT" 0.0008) ("an" 0.0008)))
("vm" (("an" 1.0)))
("wm" (("an" 1.0)))
("ym" (("nn" 0.75) ("av" 0.25)))
("äm" (("av" 0.5319) ("nn" 0.4468) ("vb" 0.0213)))
("öm" (("pm" 0.7242) ("nn" 0.2371) ("vb" 0.0284) ("av" 0.0103)))
("åm" (("nn" 1.0)))
("'n" (("nn" 0.4) ("pn" 0.2) ("**" 0.2) ("ab" 0.2)))
("-n" (("an" 0.75) ("nn" 0.25)))
("1n" (("nn" 0.5207) ("pm" 0.4684) ("**" 0.0087) ("NT" 0.0022)))
("2n" (("pm" 1.0)))
("4n" (("pm" 1.0)))
("7n" (("**" 1.0)))
(":n" (("nn" 1.0)))
("an" (("pn" 0.4237) ("nn" 0.1953) ("vb" 0.1296) ("ab" 0.0795) ("pp" 0.0772) ("kn" 0.049) ("pm" 0.0379) ("**" 0.0051) ("av" 0.0021) ("in" 0.0002) ("nl" 0.0002) ("NT" 0.0002) ("an" 0.0001)))
("bn" (("pm" 1.0)))
("cn" (("an" 1.0)))

("dn" (("an" 0.7857) ("pm" 0.2143)))
("en" (("nn" 0.4208) ("al" 0.3034) ("pn" 0.0738) ("kn" 0.0648) ("ab" 0.0525) ("pm" 0.0356) ("nl" 0.024) ("av" 0.0173) ("vb" 0.0054) ("**" 0.002) ("NT" 0.0002) ("pp" 0.0001) ("in" 0.0)))
("fn" (("an" 1.0)))
("gn" (("nn" 0.8497) ("av" 0.1088) ("**" 0.0259) ("an" 0.0104) ("pm" 0.0052)))
("hn" (("pm" 0.9392) ("nn" 0.0541) ("**" 0.0068)))
("in" (("pn" 0.5181) ("ab" 0.1957) ("pm" 0.1291) ("nn" 0.126) ("av" 0.0152) ("**" 0.0152) ("an" 0.0008)))
("ln" (("nn" 0.9267) ("pm" 0.0659) ("an" 0.0055) ("**" 0.0018)))
("mn" (("nn" 0.7444) ("pm" 0.2188) ("av" 0.0348) ("NT" 0.002)))
("nn" (("vb" 0.48) ("pm" 0.2435) ("av" 0.1217) ("nn" 0.0957) ("ab" 0.0209) ("**" 0.0209) ("pn" 0.0174)))
("on" (("nn" 0.4076) ("pn" 0.2924) ("pm" 0.2567) ("**" 0.0318) ("nl" 0.0105) ("av" 0.0005) ("NT" 0.0005) ("in" 0.0001)))
("rn" (("nn" 0.8012) ("pm" 0.1299) ("av" 0.0615) ("**" 0.0056) ("NT" 0.0019)))
("un" (("nn" 0.6162) ("pm" 0.2727) ("av" 0.0859) ("**" 0.0202) ("NT" 0.0051)))
("vn" (("NT" 0.5) ("pm" 0.5)))
("wn" (("pm" 0.7308) ("**" 0.2308) ("nn" 0.0385)))
("yn" (("nn" 0.9694) ("pm" 0.0306)))
("än" (("kn" 0.6688) ("nn" 0.1952) ("ab" 0.0915) ("av" 0.0376) ("pm" 0.007)))
("ön" (("nn" 0.6637) ("pm" 0.2183) ("av" 0.118)))
("ån" (("pp" 0.8632) ("nn" 0.0729) ("ab" 0.0565) ("pn" 0.0044) ("pm" 0.0023) ("av" 0.0004) ("vb" 0.0004)))
("/o" (("an" 1.0)))
("5o" (("pm" 0.8571) ("**" 0.1429)))
(":o" (("an" 1.0)))
("ao" (("pm" 0.6667) ("an" 0.1667) ("nn" 0.125) ("**" 0.0417)))
("bo" (("pm" 0.6886) ("vb" 0.2036) ("nn" 0.0958) ("**" 0.006) ("an" 0.006)))
("co" (("pm" 0.808) ("an" 0.112) ("**" 0.064) ("nn" 0.016)))
("do" (("pm" 0.5368) ("nn" 0.2353) ("av" 0.1471) ("**" 0.0515) ("ab" 0.0221) ("in" 0.0074)))
("eo" (("pm" 0.8571) ("nn" 0.1071) ("**" 0.0357)))
("fo" (("**" 0.5) ("nn" 0.5)))
("go" (("pm" 0.5604) ("nl" 0.1813) ("**" 0.1264) ("nn" 0.1209) ("vb" 0.0055) ("NT" 0.0055)))
("ho" (("pm" 0.7317) ("an" 0.1707) ("**" 0.0488) ("nn" 0.0244) ("pn" 0.0244)))
("io" (("nl" 0.6622) ("nn" 0.1917) ("pm" 0.1101) ("**" 0.0228) ("pn" 0.0114) ("an" 0.0019)))
("jo" (("in" 0.7468) ("pm" 0.2025) ("nn" 0.0253) ("an" 0.0253)))
("ko" (("pm" 0.5534) ("nn" 0.3592) ("**" 0.0485) ("an" 0.0291) ("vb" 0.0097)))
("lo" (("pm" 0.6154) ("nn" 0.2308) ("**" 0.0604) ("an" 0.033) ("ab" 0.0275) ("pn" 0.022) ("vb" 0.011)))
("mo" (("pm" 0.6) ("an" 0.1636) ("nn" 0.0909) ("**" 0.0909) ("ab" 0.0364) ("vb" 0.0182)))
("no" (("pm" 0.6883) ("nn" 0.1623) ("**" 0.1169) ("an" 0.0195) ("vb" 0.0065) ("pn" 0.0065)))
("oo" (("pm" 0.75) ("**" 0.25)))
("po" (("nn" 0.7143) ("pm" 0.2286) ("**" 0.0571)))
("ro" (("nn" 0.4216) ("pm" 0.305) ("vb" 0.2403) ("**" 0.0187) ("an" 0.0086) ("ab" 0.0029) ("pn" 0.0014) ("NT" 0.0014)))
("so" (("ab" 0.5652) ("pm" 0.3304) ("nn" 0.0609) ("**" 0.0435)))
("to" (("pm" 0.4) ("ab" 0.2453) ("nn" 0.1707) ("**" 0.128) ("an" 0.032) ("pn" 0.024)))
("uo" (("**" 0.8333) ("nn" 0.1667)))
("vo" (("pm" 0.9773) ("vb" 0.0227)))
("wo" (("**" 1.0)))
("xo" (("pm" 1.0)))
("yo" (("ab" 0.5333) ("pm" 0.3333) ("nn" 0.1333)))
("zo" (("pm" 0.5) ("**" 0.2778) ("nn" 0.2222)))
("ap" (("nn" 0.9491) ("pm" 0.0247) ("an" 0.0154) ("**" 0.0108)))
("bp" (("an" 1.0)))
("cp" (("an" 1.0)))
("ep" (("nn" 0.4286) ("vb" 0.3247) ("**" 0.1429) ("an" 0.0909) ("pm" 0.013)))
("fp" (("an" 1.0)))
("gp" (("an" 1.0)))
("ip" (("nn" 0.7921) ("pm" 0.1287) ("**" 0.0495) ("vb" 0.0198) ("an" 0.0099)))
("kp" (("an" 1.0)))
("lp" (("nn" 0.9757) ("vb" 0.0139) ("an" 0.0069) ("pm" 0.0035)))
("mp" (("nn" 0.8012) ("pm" 0.1739) ("**" 0.0186) ("vb" 0.0062)))
("op" (("nn" 0.4513) ("ab" 0.4351) ("**" 0.0584) ("pn" 0.0357) ("pm" 0.0097) ("an" 0.0065) ("in" 0.0032)))
("pp" (("ab" 0.5811) ("nn" 0.3769) ("pm" 0.0312) ("vb" 0.0056) ("av" 0.0043) ("in" 0.0009)))
("rp" (("pm" 0.5556) ("av" 0.2556) ("nn" 0.1) ("an" 0.0778) ("NT" 0.0111)))
("sp" (("an" 1.0)))
("tp" (("an" 1.0)))
("up" (("pm" 0.4065) ("nn" 0.3089) ("av" 0.2276) ("**" 0.0569)))
("yp" (("nn" 0.9753) ("av" 0.0185) ("pm" 0.0062)))
("äp" (("nn" 1.0)))
("öp" (("nn" 0.8246) ("vb" 0.1754)))
("åp" (("nn" 1.0)))
("sq" (("an" 1.0)))
("-r" (("an" 1.0)))
("1r" (("pm" 0.9459) ("nn" 0.0541)))
("4r" (("**" 1.0)))
("8r" (("**" 1.0)))
(":r" (("an" 1.0)))
("ar" (("vb" 0.6909) ("nn" 0.2646) ("pm" 0.0161) ("ab" 0.0113) ("av" 0.0108) ("pn" 0.0051) ("**" 0.0009) ("an" 0.0002) ("NT" 0.0)))
("br" (("an" 1.0)))
("dr" (("an" 1.0)))
("er" (("nn" 0.4154) ("vb" 0.2997) ("pp" 0.1258) ("kn" 0.0599) ("ab" 0.0439) ("pm" 0.0327) ("av" 0.0153) ("**" 0.0052) ("pn" 0.0018) ("NT" 0.0003)))
("fr" (("an" 1.0)))
("gr" (("an" 1.0)))
("hr" (("an" 0.9071) ("pm" 0.0796) ("**" 0.0133)))
("ir" (("vb" 0.8838) ("**" 0.0522) ("pm" 0.0363) ("nn" 0.0154) ("an" 0.0117) ("av" 0.0006)))
("jr" (("**" 0.5) ("an" 0.5)))
("kr" (("an" 1.0)))
("mr" (("an" 1.0)))
("nr" (("an" 1.0)))
("or" (("nn" 0.7478) ("av" 0.1217) ("vb" 0.0983) ("pm" 0.0221) ("**" 0.01) ("NT" 0.0002)))

("pr" (("an" 0.963) ("nn" 0.037)))
("rr" (("ab" 0.6266) ("nn" 0.31) ("av" 0.0262) ("pm" 0.0175) ("an" 0.0153) ("in" 0.0022) ("**" 0.0022)))
("sr" (("an" 1.0)))
("tr" (("pm" 0.5) ("an" 0.5)))
("ur" (("ab" 0.4007) ("nn" 0.3089) ("pp" 0.2412) ("pm" 0.0315) ("**" 0.0134) ("av" 0.0034) ("an" 0.001)))
("yr" (("nn" 0.4757) ("vb" 0.3784) ("av" 0.0757) ("pm" 0.0649) ("**" 0.0054)))
("är" (("vb" 0.6986) ("ab" 0.1833) ("kn" 0.0883) ("nn" 0.0217) ("av" 0.0076) ("pm" 0.0005)))
("ör" (("pp" 0.7221) ("vb" 0.1377) ("ab" 0.0963) ("nn" 0.0356) ("kn" 0.0079) ("pm" 0.0003) ("av" 0.0001)))
("år" (("vb" 0.581) ("nn" 0.3396) ("pn" 0.0692) ("av" 0.0101) ("ab" 0.0001)))
("'s" (("pm" 0.7458) ("**" 0.2542)))
("/s" (("an" 1.0)))
("1s" (("pm" 0.8947) ("**" 0.0526) ("nn" 0.0526)))
("2s" (("pm" 0.6667) ("nn" 0.3333)))
("4s" (("pm" 1.0)))
(":s" (("an" 0.8516) ("nl" 0.0801) ("nn" 0.0386) ("pm" 0.0297)))
("as" (("vb" 0.6609) ("nn" 0.1756) ("pn" 0.0721) ("pm" 0.0662) ("**" 0.0101) ("av" 0.01) ("an" 0.0041) ("ab" 0.0007) ("NT" 0.0003) ("nl" 0.0001)))
("bs" (("pm" 0.5946) ("an" 0.1892) ("nn" 0.1892) ("**" 0.027)))
("cs" (("pm" 0.4318) ("nn" 0.3636) ("**" 0.2045)))
("ds" (("pm" 0.459) ("nn" 0.2192) ("vb" 0.1895) ("ab" 0.0563) ("an" 0.0494) ("**" 0.0237) ("av" 0.003)))
("es" (("vb" 0.5831) ("pm" 0.1719) ("ab" 0.0781) ("pn" 0.0669) ("nn" 0.0471) ("**" 0.0386) ("av" 0.0139) ("pp" 0.0003)))
("fs" (("pm" 0.7) ("nn" 0.2) ("an" 0.06) ("**" 0.04)))
("gs" (("nn" 0.301) ("pm" 0.2996) ("vb" 0.2959) ("pp" 0.0631) ("av" 0.0169) ("**" 0.0103) ("ab" 0.0095) ("pn" 0.0022) ("NT" 0.0015)))
("hs" (("pm" 0.9823) ("**" 0.0088) ("nn" 0.0088)))
("is" (("ab" 0.4807) ("pm" 0.2589) ("nn" 0.2176) ("**" 0.0231) ("av" 0.0104) ("an" 0.0042) ("pn" 0.0039) ("pp" 0.0012)))
("js" (("vb" 0.6667) ("pm" 0.2727) ("nn" 0.0505) ("in" 0.0101)))
("ks" (("vb" 0.6472) ("pm" 0.2633) ("nn" 0.0584) ("**" 0.0272) ("ab" 0.0039)))
("ls" (("ab" 0.2937) ("kn" 0.2591) ("pm" 0.2041) ("vb" 0.1209) ("nn" 0.1043) ("**" 0.0173) ("an" 0.0006)))
("ms" (("pm" 0.6977) ("nn" 0.129) ("vb" 0.1205) ("an" 0.0169) ("ab" 0.0148) ("pn" 0.0127) ("**" 0.0042) ("pp" 0.0021) ("av" 0.0021)))
("ns" (("nn" 0.4991) ("vb" 0.1708) ("pn" 0.1511) ("pm" 0.1261) ("ab" 0.0458) ("**" 0.0064) ("av" 0.0003) ("NT" 0.0003) ("an" 0.0001)))
("os" (("pp" 0.6074) ("pm" 0.2298) ("nn" 0.1106) ("**" 0.0378) ("vb" 0.0071) ("nl" 0.0029) ("av" 0.0021) ("an" 0.0007) ("NT" 0.0007) ("ab" 0.0007)))
("ps" (("pm" 0.3911) ("vb" 0.2737) ("nn" 0.2458) ("**" 0.067) ("an" 0.0168) ("in" 0.0056)))
("rs" (("nn" 0.392) ("pm" 0.265) ("pn" 0.1123) ("vb" 0.0956) ("ab" 0.0808) ("**" 0.0303) ("an" 0.0171) ("av" 0.007)))
("ss" (("pn" 0.6005) ("nn" 0.1549) ("av" 0.1208) ("ab" 0.0473) ("pm" 0.0453) ("**" 0.0166) ("vb" 0.0104) ("an" 0.0037) ("in" 0.0004)))
("ts" (("vb" 0.4354) ("nn" 0.4204) ("pp" 0.0728) ("pm" 0.0654) ("**" 0.0055) ("an" 0.0003) ("pn" 0.0002)))
("us" (("nn" 0.4878) ("pm" 0.4042) ("kn" 0.0411) ("**" 0.0389) ("av" 0.0159) ("ab" 0.0101) ("nl" 0.0007) ("NT" 0.0007) ("an" 0.0007)))
("vs" (("vb" 0.6254) ("an" 0.2102) ("pm" 0.1078) ("nn" 0.0548) ("NT" 0.0018)))
("ws" (("pm" 0.6786) ("**" 0.2857) ("nn" 0.0357)))
("ys" (("pm" 0.6718) ("nn" 0.2748) ("**" 0.042) ("vb" 0.0076) ("av" 0.0038)))
("äs" (("nn" 0.7168) ("pm" 0.2301) ("vb" 0.0442) ("NT" 0.0088)))
("ös" (("av" 0.8669) ("pm" 0.0444) ("nn" 0.041) ("vb" 0.0239) ("ab" 0.0205) ("NT" 0.0034)))
("ås" (("vb" 0.4277) ("ab" 0.3295) ("pm" 0.1387) ("nn" 0.104)))
("'t" (("**" 0.75) ("pn" 0.125) ("NT" 0.125)))
(":t" (("an" 1.0)))
("at" (("vb" 0.6575) ("nn" 0.149) ("pn" 0.0988) ("av" 0.0478) ("ab" 0.0322) ("pm" 0.0087) ("**" 0.0054) ("nl" 0.0003) ("NT" 0.0001) ("kn" 0.0001)))
("bt" (("ab" 0.943) ("av" 0.0518) ("**" 0.0052)))
("ct" (("**" 0.6471) ("pm" 0.2941) ("nn" 0.0588)))
("dt" (("pm" 0.9912) ("an" 0.0088)))
("et" (("nn" 0.4425) ("pn" 0.3612) ("al" 0.0977) ("ab" 0.045) ("vb" 0.021) ("pm" 0.0187) ("av" 0.0121) ("**" 0.0017) ("an" 0.0001) ("NT" 0.0001)))
("ft" (("nn" 0.652) ("vb" 0.3208) ("pm" 0.0168) ("**" 0.0056) ("av" 0.004) ("an" 0.0008)))
("gt" (("ab" 0.4809) ("av" 0.3652) ("vb" 0.0756) ("pp" 0.0695) ("pm" 0.0077) ("pn" 0.0007) ("**" 0.0002) ("an" 0.0001)))
("ht" (("pm" 0.5403) ("**" 0.2903) ("an" 0.1048) ("av" 0.0403) ("nn" 0.0161) ("ab" 0.0081)))
("it" (("vb" 0.8379) ("ab" 0.0592) ("nn" 0.0515) ("av" 0.0216) ("pm" 0.0194) ("**" 0.0102) ("an" 0.0002)))
("jt" (("vb" 0.8462) ("av" 0.0513) ("ab" 0.0385) ("pm" 0.0385) ("nn" 0.0256)))
("kt" (("nn" 0.3154) ("ab" 0.3073) ("av" 0.2836) ("vb" 0.0846) ("pn" 0.0039) ("kn" 0.0026) ("pm" 0.0017) ("**" 0.0005) ("an" 0.0003)))
("lt" (("ab" 0.5134) ("pn" 0.1945) ("av" 0.1602) ("vb" 0.0669) ("nn" 0.0482) ("pm" 0.0148) ("**" 0.0015) ("NT" 0.0006)))
("mt" (("ab" 0.3736) ("kn" 0.2841) ("av" 0.2053) ("vb" 0.1067) ("nn" 0.0263) ("an" 0.0025) ("pm" 0.0008) ("**" 0.0008)))
("nt" (("nn" 0.3626) ("av" 0.2534) ("ab" 0.1768) ("pn" 0.0814) ("vb" 0.0623) ("pp" 0.0246) ("pm" 0.0236) ("**" 0.0153)))
("ot" (("pp" 0.467) ("pn" 0.3191) ("ab" 0.1417) ("nn" 0.0575) ("pm" 0.0126) ("**" 0.0018) ("kn" 0.0002)))
("pt" (("ab" 0.4332) ("vb" 0.3188) ("av" 0.1226) ("nn" 0.1035) ("pm" 0.0109) ("an" 0.0082) ("**" 0.0027)))
("rt" (("av" 0.3111) ("ab" 0.3089) ("vb" 0.1446) ("nn" 0.1072) ("pm" 0.063) ("pn" 0.0575) ("**" 0.0078)))
("st" (("ab" 0.6004) ("nn" 0.2017) ("av" 0.1042) ("pm" 0.0343) ("vb" 0.0257) ("kn" 0.0137)

("**" 0.0111) ("an" 0.0067) ("in" 0.001) ("pp" 0.001) ("NT" 0.0003)))
("tt" (("ie" 0.3099) ("kn" 0.2784) ("al" 0.1904) ("vb" 0.065) ("nn" 0.0538) ("pn" 0.0318) ("av" 0.0279) ("ab" 0.0239) ("nl" 0.0155) ("pm" 0.003) ("NT" 0.0003) ("**" 0.0001) ("an" 0.0001)))
("ut" (("ab" 0.7055) ("nn" 0.2041) ("NT" 0.0382) ("pm" 0.0256) ("av" 0.0193) ("**" 0.0054) ("vb" 0.0018)))
("vt" (("ab" 0.5352) ("av" 0.2781) ("vb" 0.1286) ("pn" 0.0569) ("an" 0.0012)))
("xt" (("nn" 0.7803) ("vb" 0.1364) ("av" 0.0303) ("**" 0.0227) ("an" 0.0152) ("pm" 0.0152)))
("yt" (("nn" 0.7647) ("vb" 0.2353)))
("zt" (("**" 0.5) ("pm" 0.5)))
("ät" (("vb" 0.652) ("nn" 0.2353) ("av" 0.1078) ("pm" 0.0049)))
("öt" (("vb" 0.8164) ("nn" 0.1353) ("av" 0.0435) ("**" 0.0048)))
("åt" (("pp" 0.559) ("ab" 0.2561) ("vb" 0.1084) ("nn" 0.0705) ("av" 0.0037) ("pm" 0.0015) ("pn" 0.0007)))
("au" (("**" 0.551) ("pm" 0.4184) ("an" 0.0306)))
("bu" (("pm" 0.5833) ("NT" 0.25) ("nn" 0.0833) ("in" 0.0833)))
("cu" (("pm" 1.0)))
("du" (("pn" 0.7436) ("**" 0.1987) ("pm" 0.0321) ("an" 0.0256)))
("eu" (("pm" 0.7) ("**" 0.3)))
("fu" (("pm" 0.5) ("an" 0.5)))
("gu" (("pm" 0.5) ("nl" 0.5)))
("hu" (("pm" 1.0)))
("iu" (("pm" 1.0)))
("ju" (("ab" 0.8455) ("nl" 0.1245) ("nn" 0.029) ("NT" 0.0011)))
("ku" (("pm" 0.8182) ("nn" 0.0909) ("**" 0.0909)))
("lu" (("pm" 0.6364) ("nn" 0.2727) ("**" 0.0909)))
("nu" (("ab" 0.9931) ("pm" 0.0035) ("nn" 0.0017) ("**" 0.001) ("an" 0.0007)))
("ou" (("pm" 0.7368) ("**" 0.2632)))
("ru" (("nn" 0.8793) ("kn" 0.0836) ("pm" 0.0186) ("ab" 0.0155) ("an" 0.0031)))
("su" (("pm" 0.9231) ("an" 0.0769)))
("tu" (("ab" 0.4634) ("pm" 0.2439) ("nn" 0.0976) ("**" 0.0976) ("an" 0.0732) ("nl" 0.0244)))
("uu" (("pm" 1.0)))
("zu" (("pm" 0.5) ("**" 0.5)))
("<v" (("NT" 1.0)))
("av" (("ab" 0.3455) ("vb" 0.309) ("nn" 0.2743) ("pm" 0.0693) ("pp" 0.0009) ("av" 0.0009)))
("bv" (("an" 1.0)))
("ev" (("vb" 0.8222) ("nn" 0.1328) ("pm" 0.0365) ("an" 0.0047) ("av" 0.0039)))
("iv" (("nn" 0.6594) ("av" 0.3205) ("nl" 0.0137) ("pm" 0.004) ("vb" 0.0024)))
("kv" (("an" 1.0)))
("lv" (("av" 0.8284) ("pn" 0.0826) ("nl" 0.0551) ("nn" 0.0265) ("an" 0.0032) ("pm" 0.0021) ("**" 0.0011) ("vb" 0.0011)))
("mv" (("an" 1.0)))
("ov" (("nn" 0.7484) ("pm" 0.177) ("av" 0.0341) ("vb" 0.0256) ("an" 0.0107) ("ab" 0.0043)))
("pv" (("nn" 1.0)))
("rv" (("nn" 0.9137) ("av" 0.0791) ("pm" 0.0072)))
("sv" (("an" 1.0)))
("tv" (("nn" 0.9306) ("NT" 0.0347) ("an" 0.0347)))
("uv" (("nn" 1.0)))
("xv" (("nl" 1.0)))
("yv" (("av" 1.0)))
("äv" (("nn" 0.5789) ("av" 0.3684) ("pm" 0.0526)))
("öv" (("pm" 0.6667) ("nn" 0.2381) ("vb" 0.0476) ("av" 0.0476)))
("aw" (("pm" 0.75) ("**" 0.125) ("nn" 0.125)))
("dw" (("an" 1.0)))
("ew" (("**" 0.9545) ("pm" 0.0409) ("NT" 0.0045)))
("hw" (("an" 1.0)))
("iw" (("pm" 1.0)))
("kw" (("an" 1.0)))
("mw" (("an" 1.0)))
("ow" (("pm" 0.5821) ("**" 0.2836) ("nn" 0.1343)))
("sw" (("an" 1.0)))
("uw" (("**" 1.0)))
("vw" (("pm" 1.0)))
("öw" (("pm" 1.0)))
("ax" (("ab" 0.5159) ("pm" 0.2611) ("nn" 0.2038) ("**" 0.0191)))
("cx" (("nl" 1.0)))
("ex" (("an" 0.598) ("nl" 0.2606) ("nn" 0.1005) ("pm" 0.017) ("**" 0.0119) ("NT" 0.0068) ("av" 0.0051)))
("hx" (("an" 1.0)))
("ix" (("pm" 0.6857) ("nn" 0.1429) ("**" 0.0857) ("nl" 0.0571) ("av" 0.0286)))
("nx" (("pm" 1.0)))
("ox" (("pm" 0.4) ("nn" 0.4) ("av" 0.2)))
("rx" (("pm" 1.0)))
("ux" (("pm" 0.5897) ("**" 0.2564) ("nn" 0.0769) ("ab" 0.0513) ("NT" 0.0256)))
("xx" (("nl" 1.0)))
("yx" (("nn" 1.0)))
("äx" (("nn" 1.0)))
("ay" (("pm" 0.7265) ("**" 0.2308) ("nn" 0.0427)))
("by" (("pm" 0.6683) ("nn" 0.2439) ("**" 0.0829) ("NT" 0.0049)))
("cy" (("pm" 0.5455) ("**" 0.4545)))
("dy" (("pm" 0.8729) ("**" 0.1186) ("nn" 0.0085)))
("ey" (("pm" 0.8831) ("**" 0.0779) ("nn" 0.039)))
("fy" (("pm" 0.6) ("in" 0.4)))
("gy" (("pm" 0.75) ("**" 0.25)))
("hy" (("pm" 0.8095) ("**" 0.1905)))
("iy" (("pm" 1.0)))
("ky" (("pm" 0.7468) ("nn" 0.2025) ("**" 0.0506)))
("ly" (("pm" 0.712) ("**" 0.12) ("vb" 0.104) ("nn" 0.064)))
("my" (("pm" 0.55) ("**" 0.45)))
("ny" (("av" 0.8154) ("pm" 0.1544) ("**" 0.0226) ("nn" 0.0075)))
("oy" (("pm" 0.6562) ("**" 0.2812) ("nn" 0.0625)))
("py" (("**" 0.75) ("pm" 0.25)))
("ry" (("pm" 0.5977) ("**" 0.2656) ("nn" 0.1055) ("vb" 0.0234) ("av" 0.0078)))
("sy" (("pm" 0.5) ("vb" 0.375) ("**" 0.125)))
("ty" (("kn" 0.6108) ("**" 0.2247) ("pm" 0.0759) ("nn" 0.0633) ("ab" 0.0127) ("pn" 0.0095) ("vb" 0.0032)))
("uy" (("pm" 1.0)))
("vy" (("nn" 0.76) ("pm" 0.16) ("**" 0.08)))
("wy" (("**" 1.0)))
("xy" (("an" 1.0)))
("zy" (("**" 0.6) ("pm" 0.4)))
("1z" (("pm" 1.0)))
("az" (("pm" 0.6) ("**" 0.4)))
("cz" (("pm" 1.0)))
("ez" (("pm" 0.9459) ("**" 0.0541)))
("iz" (("pm" 0.98) ("**" 0.02)))
("lz" (("pm" 1.0)))

("nz" (("pm" 0.8478) ("**" 0.087) ("nn" 0.0652)))
("oz" (("pm" 1.0)))
("rz" (("pm" 0.9231) ("**" 0.0769)))
("sz" (("pm" 1.0)))
("tz" (("pm" 0.9632) ("nn" 0.0294) ("**" 0.0074)))
("uz" (("pm" 1.0)))
("yz" (("pm" 1.0)))
("zz" (("nn" 0.6957) ("**" 0.2609) ("an" 0.0435)))
("fä" (("nn" 1.0)))
("iä" (("**" 1.0)))
("jä" (("**" 1.0)))
("kä" (("an" 1.0)))
("lä" (("vb" 0.5) ("pm" 0.3) ("nn" 0.2)))
("nä" (("nn" 0.5556) ("in" 0.4444)))
("rä" (("nn" 1.0)))
("sä" (("nn" 1.0)))
("vä" (("pm" 1.0)))
("ää" (("pm" 1.0)))
("dö" (("vb" 0.7037) ("pm" 0.2593) ("pn" 0.037)))
("eö" (("nn" 0.5) ("pm" 0.5)))
("fö" (("pm" 1.0)))
("gö" (("pm" 1.0)))
("hö" (("nn" 1.0)))
("jö" (("nn" 0.7711) ("pm" 0.1968) ("in" 0.0321)))
("kö" (("pm" 0.5172) ("nn" 0.4828)))
("lö" (("pm" 0.6667) ("av" 0.3333)))
("mö" (("pm" 0.9565) ("nn" 0.0386) ("NT"
0.0048)))
("nö" (("nn" 0.7714) ("pm" 0.2286)))
("pö" (("nn" 1.0)))
("rö" (("pm" 0.5455) ("nn" 0.3636) ("vb" 0.0909)))
("sö" (("pm" 0.8571) ("NT" 0.0714) ("an" 0.0714)))
("tö" (("nn" 1.0)))
("vö" (("pm" 0.75) ("nn" 0.25)))
("öö" (("pm" 1.0)))
("då" (("ab" 0.6624) ("kn" 0.3344) ("in" 0.0023)
("nn" 0.0009)))
("eå" (("pm" 1.0)))
("få" (("vb" 0.8632) ("av" 0.1353) ("nn" 0.0014)))
("gå" (("vb" 0.9985) ("**" 0.0015)))
("hå" (("nn" 1.0)))
("kå" (("pm" 1.0)))
("lå" (("vb" 0.6091) ("av" 0.3621) ("nn" 0.0247)
("pm" 0.0041)))
("må" (("av" 0.7) ("vb" 0.2946) ("**" 0.0054)))
("nå" (("vb" 0.8235) ("in" 0.1471) ("pm" 0.0221)
("nn" 0.0074)))
("på" (("pp" 0.9782) ("ab" 0.0212) ("nn" 0.0004)
("vb" 0.0002)))
("rå" (("nn" 0.5044) ("av" 0.4336) ("pm" 0.0265)
("vb" 0.0265) ("ab" 0.0088)))
("så" (("ab" 0.9665) ("pn" 0.0162) ("kn" 0.0158)
("vb" 0.0009) ("in" 0.0006)))
("tå" (("vb" 0.9784) ("nn" 0.0216)))
("vå" (("nl" 0.9198) ("nn" 0.0795) ("vb" 0.0008)))

THREE LETTER ENDING LEXICON (EXCERPT)

```
("ank" (("nn" 0.4684) ("pm" 0.3418) ("**" 0.1013)
    ("av" 0.0759) ("vb" 0.0127)))
("enk" (("**" 1.0)))
("ink" (("nn" 0.6071) ("pm" 0.3214) ("**" 0.0714)))
("onk" (("pm" 1.0)))
("unk" (("nn" 1.0)))
("änk" (("vb" 0.5075) ("nn" 0.4925)))
("önk" (("vb" 1.0)))
("bok" (("nn" 0.9942) ("pm" 0.0058)))
("gok" (("nn" 1.0)))
("kok" (("pm" 0.625) ("nn" 0.375)))
("lok" (("av" 0.8261) ("nn" 0.1739)))
("nok" (("pm" 0.5) ("**" 0.5)))
("ook" (("pm" 0.7179) ("**" 0.2564) ("nn"
    0.0256)))
("pok" (("nn" 1.0)))
("rok" (("nn" 1.0)))
("tok" (("nn" 1.0)))
("spk" (("an" 1.0)))
("8rk" (("pm" 0.5) ("**" 0.5)))
("ark" (("pm" 0.4216) ("nn" 0.2903) ("av" 0.2712)
    ("**" 0.0169)))
("erk" (("nn" 0.9902) ("**" 0.0059) ("pm" 0.0039)))
("frk" (("an" 1.0)))
("irk" (("pm" 1.0)))
("ork" (("pm" 0.9104) ("nn" 0.0672) ("**" 0.0224)))
("urk" (("nn" 1.0)))
("ärk" (("nn" 0.7273) ("vb" 0.2727)))
("örk" (("av" 0.8286) ("pm" 0.1143) ("nn" 0.0571)))
("ask" (("nn" 0.6757) ("av" 0.2162) ("**" 0.0541)
    ("pm" 0.027) ("ab" 0.027)))
("dsk" (("av" 1.0)))
("esk" (("av" 0.7826) ("nn" 0.2174)))
("gsk" (("av" 1.0)))
("isk" (("av" 0.9466) ("nn" 0.0519) ("pm" 0.0015)))
("lsk" (("av" 1.0)))
("msk" (("av" 0.7727) ("an" 0.2273)))
("nsk" (("av" 0.9566) ("nn" 0.0364) ("**" 0.0056)
    ("pm" 0.0014)))
("osk" (("nn" 1.0)))
("psk" (("av" 1.0)))
("rsk" (("av" 0.8548) ("nn" 0.0806) ("pm" 0.0484)
    ("**" 0.0161)))
("tsk" (("an" 0.6) ("av" 0.4)))
("usk" (("nn" 0.5) ("pm" 0.5)))
("wsk" (("av" 1.0)))
("ysk" (("av" 0.9298) ("nn" 0.0702)))
("äsk" (("nn" 0.9565) ("pm" 0.0435)))
("åsk" (("nn" 1.0)))
("ltk" (("an" 1.0)))
("buk" (("nn" 1.0)))
("duk" (("nn" 1.0)))
("euk" (("pm" 1.0)))
("juk" (("av" 1.0)))
("ouk" (("pm" 1.0)))
("puk" (("nn" 1.0)))
("ruk" (("nn" 0.9942) ("pm" 0.0058)))
("tuk" (("pm" 1.0)))
("awk" (("pm" 1.0)))
("ryk" (("pm" 0.5556) ("nn" 0.4444)))
("tyk" (("nn" 1.0)))
("zyk" (("pm" 1.0)))
("käk" (("nn" 1.0)))
("dök" (("vb" 1.0)))
("gök" (("pm" 1.0)))
("hök" (("nn" 1.0)))
("kök" (("nn" 1.0)))
("lök" (("nn" 1.0)))
("rök" (("vb" 0.7234) ("nn" 0.2128) ("pm" 0.0638)))
("sök" (("nn" 0.9777) ("vb" 0.0199) ("pm"
    0.0025)))
("öök" (("pm" 1.0)))
("kåk" (("nn" 1.0)))
("låk" (("nn" 1.0)))
("påk" (("nn" 1.0)))
("råk" (("nn" 1.0)))
("xåk" (("nn" 1.0)))
("e4l" (("pm" 1.0)))
("1al" (("pm" 1.0)))
("aal" (("pm" 1.0)))
("bal" (("pm" 0.8889) ("av" 0.1111)))
("cal" (("**" 0.56) ("nn" 0.32) ("an" 0.12)))
("dal" (("pm" 0.6575) ("nn" 0.274) ("av" 0.0548)
    ("**" 0.0137)))
("eal" (("nn" 0.8302) ("pm" 0.1132) ("**" 0.0377)
    ("av" 0.0189)))
("fal" (("pm" 0.6) ("av" 0.4)))
("gal" (("pm" 0.8537) ("av" 0.1463)))
("hal" (("pm" 0.5455) ("av" 0.2727) ("**" 0.0909)
    ("nn" 0.0909)))
("ial" (("nn" 0.6969) ("av" 0.2312) ("**" 0.0688)
    ("pm" 0.0031)))
("jal" (("av" 1.0)))
("kal" (("nn" 0.5876) ("av" 0.4021) ("**" 0.0103)))
("mal" (("av" 0.9494) ("vb" 0.0211) ("**" 0.0169)
    ("nn" 0.0127)))
("nal" (("nn" 0.7233) ("**" 0.1311) ("av" 0.1311)
    ("pm" 0.0146)))
("oal" (("nn" 1.0)))
("pal" (("nn" 0.5) ("pm" 0.5)))
("ral" (("nn" 0.5816) ("av" 0.3231) ("**" 0.0714)
    ("pm" 0.0238)))
("sal" (("nn" 0.8906) ("av" 0.0625) ("pm" 0.0312)
    ("**" 0.0156)))
("tal" (("nn" 0.8698) ("av" 0.0991) ("**" 0.0127)
    ("pm" 0.0115) ("vb" 0.0069)))
("ual" (("nn" 0.5714) ("**" 0.2857) ("pm" 0.1429)))
("val" (("nn" 0.9286) ("pm" 0.0556) ("av" 0.0079)
    ("**" 0.0079)))
("xal" (("av" 1.0)))
("yal" (("**" 0.875) ("pm" 0.125)))
("sbl" (("an" 1.0)))
("ddl" (("an" 1.0)))
("ödl" (("pm" 1.0)))
("ael" (("pm" 1.0)))
("bel" (("av" 0.5254) ("nn" 0.2627) ("pm" 0.1186)
    ("pn" 0.0932)))
("cel" (("pm" 0.8889) ("nn" 0.1111)))
("del" (("nn" 0.9351) ("pm" 0.0396) ("ab" 0.0144)
    ("**" 0.0062) ("av" 0.0048)))
("eel" (("**" 0.8) ("pm" 0.2)))
("fel" (("nn" 0.4513) ("av" 0.2974) ("ab" 0.2462)
    ("NT" 0.0051)))
("gel" (("nn" 0.9611) ("pm" 0.0311) ("**" 0.0078)))
("hel" (("av" 0.9112) ("pm" 0.0888)))
("iel" (("nn" 0.5778) ("pm" 0.4222)))
("jel" (("pm" 0.6667) ("nn" 0.3333)))
("kel" (("nn" 0.8042) ("av" 0.1632) ("pm" 0.0297)
    ("**" 0.003)))
("mel" (("nn" 0.7255) ("pm" 0.2745)))
("nel" (("pm" 0.7391) ("nn" 0.2174) ("**" 0.0435)))
("oel" (("pm" 1.0)))
("pel" (("nn" 0.9656) ("pm" 0.0275) ("**" 0.0052)
    ("av" 0.0017)))
```

("rel" (("pm" 1.0)))
("sel" (("nn" 0.6441) ("pm" 0.2938) ("av" 0.0565)
("**" 0.0056)))
("tel" (("nn" 0.7697) ("**" 0.125) ("av" 0.0526)
("pm" 0.0329) ("an" 0.0197)))
("uel" (("pm" 1.0)))
("vel" (("nn" 0.8661) ("**" 0.0804) ("pm" 0.0536)))
("wel" (("pm" 1.0)))
("xel" (("pm" 0.8548) ("nn" 0.1452)))
("zel" (("pm" 0.8571) ("nn" 0.1429)))
("igl" (("pm" 1.0)))
("ngl" (("an" 1.0)))
("ahl" (("pm" 1.0)))
("ihl" (("pm" 1.0)))
("ohl" (("pm" 1.0)))
("uhl" (("pm" 1.0)))
("öhl" (("pm" 1.0)))
("åhl" (("pm" 1.0)))
("ail" (("pm" 0.8) ("nn" 0.2)))
("bil" (("nn" 0.9408) ("av" 0.0592)))
("cil" (("**" 0.5333) ("pm" 0.2667) ("av" 0.2)))
("eil" (("**" 1.0)))
("fil" (("nn" 0.6) ("an" 0.4)))
("gil" (("pm" 1.0)))
("hil" (("pm" 1.0)))
("kil" (("pm" 0.9) ("nn" 0.1)))
("lil" (("pm" 1.0)))
("mil" (("nn" 0.8451) ("pm" 0.1549)))
("oil" (("**" 1.0)))
("pil" (("nn" 1.0)))
("ril" (("nn" 0.9172) ("av" 0.0621) ("NT" 0.0138)
("pm" 0.0069)))
("sil" (("pm" 1.0)))
("til" (("nn" 0.724) ("pm" 0.2448) ("av" 0.0312)))
("uil" (("pm" 1.0)))
("vil" (("av" 0.625) ("**" 0.375)))
("xil" (("nn" 1.0)))
("zil" (("pm" 1.0)))
("xkl" (("an" 1.0)))
("'ll" (("**" 1.0)))
("1ll" (("pm" 1.0)))
("all" (("vb" 0.5966) ("nn" 0.2637) ("pn" 0.0942)
("pm" 0.0238) ("av" 0.0083) ("kn" 0.0074)
("**" 0.0056) ("ab" 0.0003)))
("ell" (("av" 0.5487) ("pm" 0.2273) ("nn" 0.2161)
("**" 0.0067) ("in" 0.0011)))
("ill" (("pp" 0.8557) ("vb" 0.0754) ("ab" 0.0564)
("pm" 0.0077) ("nn" 0.0035) ("av" 0.0007)
("**" 0.0005) ("NT" 0.0001)))
("jll" (("pm" 1.0)))
("oll" (("nn" 0.9606) ("pm" 0.0157) ("**" 0.0105)
("nl" 0.0105) ("an" 0.0026)))
("ull" (("av" 0.6465) ("nn" 0.3131) ("pm" 0.0202)
("ab" 0.0177) ("**" 0.0025)))
("yll" (("nn" 0.931) ("vb" 0.0345) ("pm" 0.0345)))
("äll" (("nn" 0.7983) ("av" 0.1429) ("vb" 0.0336)
("pm" 0.0252)))
("öll" (("vb" 0.9915) ("pm" 0.0085)))
("åll" (("nn" 0.9958) ("vb" 0.0042)))
("eml" (("pm" 1.0)))
("dnl" (("an" 1.0)))
("bol" (("nn" 1.0)))
("dol" (("nn" 1.0)))
("eol" (("an" 1.0)))
("fol" (("nn" 1.0)))
("gol" (("pm" 0.5) ("vb" 0.5)))
("hol" (("nn" 0.8) ("pm" 0.2)))
("iol" (("nn" 1.0)))
("jol" (("nn" 1.0)))
("kol" (("nn" 0.875) ("an" 0.125)))
("nol" (("pm" 0.6667) ("nn" 0.3333)))
("ool" (("**" 0.5294) ("nn" 0.2941) ("pm" 0.1765)))
("pol" (("nn" 0.8421) ("pm" 0.1053) ("**" 0.0526)))
("rol" (("pm" 0.6) ("nn" 0.3) ("**" 0.1)))
("sol" (("nn" 0.7209) ("**" 0.2093) ("pm" 0.0698)))
("tol" (("nn" 0.9375) ("pm" 0.0625)))
("vol" (("an" 1.0)))
("zol" (("nn" 1.0)))
("arl" (("pm" 0.8657) ("nn" 0.1157) ("**" 0.0185)))
("irl" (("**" 1.0)))
("orl" (("nn" 1.0)))
("ärl" (("nn" 1.0)))
("aul" (("pm" 0.9836) ("NT" 0.0164)))
("bul" (("pm" 0.8571) ("nn" 0.1429)))
("dul" (("**" 1.0)))
("eul" (("pm" 1.0)))
("ful" (("av" 0.7778) ("**" 0.2222)))
("gul" (("av" 1.0)))
("iul" (("**" 1.0)))
("jul" (("nn" 1.0)))
("kul" (("av" 0.9) ("pm" 0.1)))
("oul" (("pm" 0.8889) ("**" 0.1111)))
("sul" (("nn" 1.0)))
("owl" (("**" 1.0)))
("sxl" (("an" 1.0)))
("byl" (("pm" 0.6667) ("nn" 0.3333)))
("dyl" (("an" 0.8333) ("nn" 0.1667)))
("kyl" (("nn" 1.0)))
("nyl" (("nn" 1.0)))
("ryl" (("nn" 1.0)))
("syl" (("nn" 1.0)))
("tyl" (("nn" 1.0)))
("fäl" (("nn" 1.0)))
("gäl" (("nn" 1.0)))
("häl" (("nn" 1.0)))
("jäl" (("nn" 0.4098) ("ab" 0.2787) ("av" 0.2459)
("vb" 0.0656)))
("käl" (("nn" 1.0)))
("räl" (("nn" 1.0)))
("väl" (("ab" 0.8656) ("kn" 0.118) ("nn" 0.0113)
("in" 0.0052)))
("föl" (("nn" 1.0)))
("jöl" (("nn" 1.0)))
("köl" (("nn" 1.0)))
("löl" (("nn" 0.6667) ("pm" 0.3333)))
("nöl" (("nn" 1.0)))
("pöl" (("nn" 1.0)))
("röl" (("**" 1.0)))
("yöl" (("nn" 1.0)))
("bål" (("nn" 1.0)))
("hål" (("nn" 1.0)))
("kål" (("nn" 0.9) ("pm" 0.1)))

FOUR LETTER ENDING LEXICON (EXCERPT)

```
("ball" (("nn" 0.5) ("pm" 0.5)))
("call" (("pm" 1.0)))
("dall" (("pm" 1.0)))
("fall" (("nn" 0.9665) ("kn" 0.0299) ("pm" 0.0012)
    ("ab" 0.0012) ("vb" 0.0012)))
("gall" (("pm" 1.0)))
("hall" (("pm" 0.449) ("nn" 0.4082) ("**" 0.1429)))
("kall" (("vb" 0.9809) ("av" 0.0137) ("nn" 0.0049)
    ("pm" 0.0005)))
("mall" (("nn" 0.6667) ("vb" 0.1111) ("**" 0.1111)
    ("pm" 0.1111)))
("nall" (("nn" 1.0)))
("pall" (("nn" 0.8) ("pm" 0.2)))
("rall" (("nn" 1.0)))
("tall" (("nn" 1.0)))
("vall" (("pm" 0.6667) ("nn" 0.3333)))
("wall" (("pm" 0.875) ("**" 0.125)))
("yall" (("pm" 1.0)))
("bell" (("nn" 0.625) ("pm" 0.375)))
("cell" (("nn" 1.0)))
("dell" (("nn" 0.6304) ("pm" 0.3696)))
("eell" (("av" 1.0)))
("gell" (("pm" 1.0)))
("hell" (("**" 0.4) ("pm" 0.4) ("in" 0.2)))
("iell" (("av" 0.9914) ("nn" 0.0086)))
("jell" (("pm" 1.0)))
("kell" (("pm" 1.0)))
("lell" (("nn" 0.7143) ("av" 0.2857)))
("mell" (("av" 0.7647) ("nn" 0.1765) ("pm"
    0.0588)))
("nell" (("av" 0.8502) ("pm" 0.1498)))
("pell" (("nn" 1.0)))
("rell" (("pm" 0.6905) ("av" 0.2381) ("nn" 0.0714)))
("sell" (("pm" 0.8438) ("nn" 0.0938) ("av"
    0.0625)))
("tell" (("nn" 0.7449) ("pm" 0.1837) ("av" 0.0408)
    ("**" 0.0306)))
("uell" (("av" 1.0)))
("vell" (("nn" 1.0)))
("well" (("pm" 0.9643) ("**" 0.0357)))
("xell" (("pm" 1.0)))
("zell" (("pm" 1.0)))
("bill" (("pm" 1.0)))
("cill" (("nn" 1.0)))
("dill" (("nn" 1.0)))
("eill" (("pm" 1.0)))
("gill" (("av" 0.5) ("nn" 0.3333) ("pm" 0.1667)))
("hill" (("pm" 0.9492) ("**" 0.0339) ("NT"
    0.0169)))
("lill" (("pm" 1.0)))
("mill" (("pm" 1.0)))
("pill" (("nn" 1.0)))
("rill" (("nn" 0.75) ("pm" 0.25)))
("sill" (("nn" 1.0)))
("till" (("pp" 0.9379) ("ab" 0.0619) ("nn" 0.0001)
    ("**" 0.0001)))
("vill" (("vb" 0.9954) ("av" 0.0023) ("pm" 0.0011)
    ("**" 0.0011)))
("will" (("nn" 0.6) ("pm" 0.3) ("**" 0.1)))
("ejll" (("pm" 1.0)))
("boll" (("nn" 1.0)))
("doll" (("pm" 0.5) ("an" 0.5)))
("koll" (("nn" 0.75) ("pm" 0.25)))
("moll" (("nn" 1.0)))
("noll" (("nl" 1.0)))
("roll" (("nn" 0.9852) ("**" 0.0119) ("pm" 0.003)))
("soll" (("nn" 1.0)))
("toll" (("pm" 1.0)))
("bull" (("pm" 1.0)))
("full" (("av" 0.9624) ("nn" 0.0338) ("**" 0.0038)))
("hull" (("nn" 1.0)))
("kull" (("nn" 0.9082) ("ab" 0.0714) ("pm"
    0.0204)))
("mull" (("nn" 1.0)))
("null" (("nn" 1.0)))
("rull" (("nn" 1.0)))
("tull" (("nn" 0.6667) ("pm" 0.3333)))
("dyll" (("nn" 1.0)))
("fyll" (("nn" 1.0)))
("kyll" (("vb" 0.5) ("pm" 0.5)))
("tyll" (("nn" 1.0)))
("fäll" (("nn" 1.0)))
("gäll" (("av" 1.0)))
("häll" (("pm" 0.4) ("vb" 0.4) ("nn" 0.2)))
("jäll" (("nn" 0.9091) ("pm" 0.0909)))
("käll" (("nn" 1.0)))
("mäll" (("nn" 1.0)))
("näll" (("av" 0.7778) ("nn" 0.2222)))
("räll" (("nn" 0.5) ("av" 0.5)))
("täll" (("nn" 0.6) ("vb" 0.4)))
("väll" (("nn" 1.0)))
("böll" (("pm" 1.0)))
("föll" (("vb" 1.0)))
("höll" (("vb" 1.0)))
("fåll" (("nn" 1.0)))
("håll" (("nn" 0.9958) ("vb" 0.0042)))
("reml" (("pm" 1.0)))
("mbol" (("nn" 1.0)))
("idol" (("nn" 1.0)))
("teol" (("an" 1.0)))
("ifol" (("nn" 1.0)))
("ohol" (("nn" 0.8) ("pm" 0.2)))
("fiol" (("nn" 1.0)))
("riol" (("nn" 1.0)))
("viol" (("nn" 1.0)))
("fjol" (("nn" 1.0)))
("kjol" (("nn" 1.0)))
("skol" (("an" 1.0)))
("äkol" (("nn" 1.0)))
("anol" (("nn" 1.0)))
("enol" (("nn" 1.0)))
("inol" (("pm" 1.0)))
("cool" (("**" 1.0)))
("hool" (("**" 1.0)))
("pool" (("nn" 0.5556) ("pm" 0.3333) ("**"
    0.1111)))
("opol" (("nn" 1.0)))
("rpol" (("pm" 1.0)))
("tpol" (("nn" 1.0)))
("arol" (("pm" 1.0)))
("erol" (("nn" 1.0)))
("frol" (("nn" 1.0)))
("rrol" (("pm" 1.0)))
("trol" (("**" 1.0)))
("esol" (("nn" 1.0)))
("isol" (("pm" 1.0)))
("nsol" (("nn" 1.0)))
("rsol" (("nn" 1.0)))
("ssol" (("nn" 1.0)))
("itol" (("pm" 1.0)))
("stol" (("nn" 0.9524) ("pm" 0.0476)))
("azol" (("nn" 1.0)))
("carl" (("pm" 1.0)))
("earl" (("**" 0.5) ("pm" 0.25) ("nn" 0.25)))
```

("jarl" (("pm" 1.0)))
("karl" (("pm" 0.8189) ("nn" 0.1811)))
("girl" (("**" 1.0)))
("sorl" (("nn" 1.0)))
("kärl" (("nn" 1.0)))
("paul" (("pm" 0.9836) ("NT" 0.0164)))
("abul" (("pm" 1.0)))
("ibul" (("nn" 1.0)))
("nbul" (("pm" 1.0)))
("rdul" (("**" 1.0)))
("reul" (("pm" 1.0)))
("nful" (("**" 1.0)))
("pful" (("**" 1.0)))
("dgul" (("av" 1.0)))
("egul" (("av" 1.0)))
("lgul" (("av" 1.0)))
("ågul" (("av" 1.0)))
("aiul" (("**" 1.0)))
("hjul" (("nn" 1.0)))
("kjul" (("nn" 1.0)))
("tkul" (("pm" 1.0)))
("aoul" (("pm" 1.0)))
("poul" (("pm" 1.0)))
("soul" (("**" 1.0)))
("nsul" (("nn" 1.0)))
("fowl" (("**" 1.0)))
("abyl" (("nn" 1.0)))
("ibyl" (("pm" 1.0)))
("ndyl" (("nn" 1.0)))
("ekyl" (("nn" 1.0)))
("lkyl" (("nn" 1.0)))
("inyl" (("nn" 1.0)))
("cryl" (("nn" 1.0)))
("pryl" (("nn" 1.0)))
("asyl" (("nn" 1.0)))
("ntyl" (("nn" 1.0)))
("efäl" (("nn" 1.0)))
("shäl" (("nn" 1.0)))
("ejäl" (("av" 1.0)))
("hjäl" (("ab" 1.0)))
("själ" (("nn" 1.0)))
("tjäl" (("vb" 1.0)))
("skäl" (("nn" 1.0)))
("gräl" (("nn" 1.0)))
("aväl" (("ab" 1.0)))
("kväl" (("ab" 1.0)))
("mväl" (("ab" 1.0)))
("rväl" (("nn" 1.0)))
("åväl" (("kn" 0.9577) ("in" 0.0423)))
("mjöl" (("nn" 1.0)))
("rköl" (("nn" 1.0)))
("elöl" (("pm" 1.0)))
("ulöl" (("nn" 1.0)))
("anöl" (("nn" 1.0)))
("knöl" (("nn" 1.0)))
("dpöl" (("nn" 1.0)))
("oröl" (("**" 1.0)))
("byöl" (("nn" 1.0)))
("rbål" (("nn" 1.0)))
("sbål" (("nn" 1.0)))
("lhål" (("nn" 1.0)))
("nhål" (("nn" 1.0)))
("phål" (("nn" 1.0)))
("shål" (("nn" 1.0)))
("thål" (("nn" 1.0)))
("dkål" (("nn" 1.0)))
("lkål" (("nn" 1.0)))
("mkål" (("nn" 1.0)))
("skål" (("nn" 0.8421) ("pm" 0.1579)))
("tkål" (("nn" 1.0)))
("amål" (("nn" 1.0)))
("bmål" (("nn" 1.0)))
("dmål" (("nn" 1.0)))
("emål" (("nn" 1.0)))
("nmål" (("nn" 1.0)))
("omål" (("nn" 1.0)))
("pmål" (("nn" 1.0)))
("rmål" (("nn" 1.0)))
("smål" (("nn" 0.9831) ("**" 0.0169)))
("tmål" (("nn" 1.0)))
("vmål" (("nn" 1.0)))
("åmål" (("pm" 1.0)))
("rnål" (("nn" 1.0)))
("snål" (("av" 1.0)))
("tnål" (("nn" 1.0)))
("trål" (("nn" 1.0)))
("vrål" (("nn" 1.0)))
("otål" (("nn" 1.0)))
("stål" (("nn" 0.9) ("pm" 0.1)))
("svål" (("nn" 1.0)))
("tvål" (("nn" 1.0)))
("he3m" (("nn" 1.0)))
("laam" (("pm" 1.0)))
("ocam" (("an" 1.0)))
("adam" (("pm" 0.8125) ("nn" 0.1875)))
("ldam" (("nn" 1.0)))
("rdam" (("pm" 1.0)))
("sdam" (("pm" 0.5) ("nn" 0.5)))
("beam" (("pm" 1.0)))
("ream" (("**" 1.0)))
("team" (("nn" 0.9167) ("**" 0.0833)))
("aham" (("pm" 1.0)))
("dham" (("pm" 1.0)))
("eham" (("pm" 1.0)))
("gham" (("pm" 1.0)))
("kham" (("pm" 1.0)))
("lham" (("pm" 1.0)))
("pham" (("pm" 1.0)))
("sham" (("pm" 1.0)))
("tham" (("pm" 1.0)))
("ciam" (("**" 1.0)))
("liam" (("pm" 1.0)))
("siam" (("pm" 1.0)))
("skam" (("nn" 1.0)))
("glam" (("nn" 1.0)))
("klam" (("nn" 1.0)))
("slam" (("nn" 1.0)))
("tnam" (("pm" 1.0)))
("noam" (("pm" 1.0)))
("dram" (("nn" 1.0)))
("fram" (("ab" 0.9988) ("pm" 0.0012)))
("gram" (("nn" 0.9793) ("**" 0.0124) ("pm"
0.0083)))
("iram" (("pm" 0.7143) ("nn" 0.2857)))
("jram" (("nn" 1.0)))
("sram" (("nn" 1.0)))
("tram" (("av" 0.8889) ("nn" 0.1111)))
("dsam" (("av" 1.0)))
("esam" (("nn" 0.5) ("av" 0.5)))
("gsam" (("av" 1.0)))
("jsam" (("av" 1.0)))
("ksam" (("av" 0.9877) ("pm" 0.0123)))
("lsam" (("av" 0.9412) ("nn" 0.0588)))
("nsam" (("av" 1.0)))
("osam" (("av" 1.0)))

FIVE LETTER ENDING LEXICON (EXCERPT)

```
("legal" (("av" 1.0)))
("negal" (("pm" 1.0)))
("segal" (("pm" 1.0)))
("tigal" (("pm" 1.0)))
("tugal" (("pm" 1.0)))
("schal" (("**" 0.5) ("nn" 0.5)))
("nthal" (("pm" 1.0)))
("ecial" (("**" 1.0)))
("icial" (("**" 1.0)))
("ncial" (("**" 1.0)))
("ocial" (("av" 0.9259) ("**" 0.0741)))
("egial" (("av" 1.0)))
("ilial" (("nn" 0.875) ("**" 0.125)))
("enial" (("av" 1.0)))
("onial" (("**" 0.7143) ("av" 0.2857)))
("erial" (("nn" 0.9769) ("**" 0.0231)))
("orial" (("nn" 1.0)))
("trial" (("**" 1.0)))
("asial" (("av" 1.0)))
("esial" (("pm" 1.0)))
("ntial" (("nn" 1.0)))
("ivial" (("av" 1.0)))
("lojal" (("av" 1.0)))
("dikal" (("av" 0.9615) ("nn" 0.0385)))
("rikal" (("av" 1.0)))
("tikal" (("nn" 0.5) ("av" 0.5)))
("lokal" (("nn" 0.7317) ("av" 0.2683)))
("pokal" (("nn" 1.0)))
("vokal" (("nn" 1.0)))
("iskal" (("nn" 1.0)))
("nskal" (("nn" 1.0)))
("tskal" (("nn" 1.0)))
("gamal" (("av" 1.0)))
("admal" (("nn" 1.0)))
("nimal" (("av" 0.8333) ("**" 0.1667)))
("timal" (("av" 1.0)))
("ximal" (("av" 1.0)))
("nkmal" (("**" 1.0)))
("ammal" (("av" 1.0)))
("ormal" (("av" 1.0)))
("esmal" (("av" 1.0)))
("gsmal" (("av" 1.0)))
("lsmal" (("av" 1.0)))
("rsmal" (("nn" 1.0)))
("zumal" (("**" 1.0)))
("banal" (("av" 1.0)))
("canal" (("pm" 1.0)))
("kanal" (("nn" 1.0)))
("menal" (("av" 1.0)))
("senal" (("nn" 1.0)))
("ignal" (("nn" 0.875) ("pm" 0.125)))
("dinal" (("nn" 1.0)))
("final" (("nn" 1.0)))
("ginal" (("nn" 0.8571) ("**" 0.1429)))
("minal" (("nn" 1.0)))
("ennal" (("nn" 1.0)))
("gonal" (("av" 1.0)))
("ional" (("**" 0.9231) ("av" 0.0769)))
("sonal" (("nn" 1.0)))
("urnal" (("nn" 1.0)))
("bunal" (("nn" 1.0)))
("munal" (("av" 1.0)))
("spoal" (("nn" 1.0)))
("nepal" (("pm" 1.0)))
("cipal" (("nn" 1.0)))
("ebral" (("av" 1.0)))
("edral" (("nn" 1.0)))
("odral" (("nn" 1.0)))
("beral" (("av" 0.8571) ("nn" 0.0952) ("**"
    0.0476)))
("deral" (("av" 0.6471) ("**" 0.3529)))
("neral" (("nn" 0.9277) ("**" 0.0723)))
("teral" (("av" 1.0)))
("miral" (("nn" 1.0)))
("akral" (("av" 1.0)))
("skral" (("av" 1.0)))
("coral" (("**" 1.0)))
("koral" (("nn" 1.0)))
("moral" (("nn" 1.0)))
("toral" (("nn" 0.6667) ("**" 0.3333)))
("rpral" (("nn" 1.0)))
("orral" (("**" 1.0)))
("ntral" (("av" 0.7358) ("nn" 0.2264) ("**"
    0.0377)))
("stral" (("av" 1.0)))
("utral" (("av" 1.0)))
("tural" (("av" 0.625) ("**" 0.375)))
("tesal" (("nn" 1.0)))
("eisal" (("nn" 1.0)))
("sisal" (("nn" 1.0)))
("iksal" (("nn" 1.0)))
("alsal" (("nn" 1.0)))
("elsal" (("nn" 1.0)))
("ansal" (("pm" 1.0)))
("onsal" (("nn" 1.0)))
("rnsal" (("nn" 1.0)))
("ersal" (("**" 1.0)))
("ärsal" (("nn" 1.0)))
("örsal" (("nn" 1.0)))
("assal" (("pm" 1.0)))
("dssal" (("nn" 1.0)))
("gssal" (("nn" 1.0)))
("nssal" (("nn" 1.0)))
("ossal" (("av" 1.0)))
("tssal" (("nn" 1.0)))
("atsal" (("nn" 1.0)))
("ktsal" (("nn" 1.0)))
("rtsal" (("nn" 1.0)))
("stsal" (("nn" 1.0)))
("ttsal" (("nn" 1.0)))
("övsal" (("nn" 1.0)))
("f-tal" (("nn" 1.0)))
("n-tal" (("nn" 1.0)))
("v-tal" (("nn" 1.0)))
("fatal" (("av" 1.0)))
("natal" (("pm" 1.0)))
("ratal" (("nn" 1.0)))
("adtal" (("nn" 1.0)))
("ndtal" (("nn" 1.0)))
("ietal" (("nn" 1.0)))
("retal" (("nn" 1.0)))
("setal" (("nn" 1.0)))
("tetal" (("nn" 1.0)))
("ngtal" (("nn" 1.0)))
("nital" (("av" 1.0)))
("pital" (("nn" 0.8723) ("**" 0.1277)))
("vital" (("av" 1.0)))
("cktal" (("nn" 1.0)))
("altal" (("nn" 1.0)))
("eltal" (("nn" 1.0)))
("lltal" (("nn" 1.0)))
("åltal" (("nn" 1.0)))
("amtal" (("nn" 1.0)))
("emtal" (("nn" 1.0)))
("antal" (("nn" 0.9885) ("pm" 0.0115)))
```

("ental" (("av" 0.6744) ("nn" 0.2791) ("**" 0.0465)))
("intal" (("nn" 1.0)))
("ontal" (("nn" 0.8) ("av" 0.2)))
("gotal" (("nn" 1.0)))
("iotal" (("nn" 1.0)))
("total" (("av" 0.9412) ("**" 0.0588)))
("artal" (("nn" 1.0)))
("ertal" (("nn" 0.9) ("pm" 0.1)))
("ortal" (("pm" 1.0)))
("örtal" (("nn" 1.0)))
("dstal" (("nn" 1.0)))
("estal" (("nn" 1.0)))
("gstal" (("nn" 1.0)))
("nstal" (("nn" 1.0)))
("rstal" (("nn" 0.8333) ("pm" 0.1667)))
("ystal" (("**" 1.0)))
("sttal" (("nn" 1.0)))
("uttal" (("nn" 1.0)))
("rutal" (("av" 0.75) ("nn" 0.125) ("pm" 0.125)))
("avtal" (("nn" 1.0)))
("ovtal" (("nn" 1.0)))
("rvtal" (("nn" 1.0)))
("fåtal" (("nn" 1.0)))
("såtal" (("nn" 1.0)))
("våtal" (("nn" 1.0)))
("scual" (("pm" 1.0)))
("idual" (("**" 1.0)))
("itual" (("nn" 1.0)))
("utual" (("**" 1.0)))
("e-val" (("nn" 1.0)))
("m-val" (("nn" 1.0)))
("laval" (("pm" 1.0)))
("naval" (("nn" 1.0)))
("raval" (("nn" 1.0)))
("ldval" (("nn" 1.0)))
("ndval" (("nn" 1.0)))
("rdval" (("nn" 1.0)))
("neval" (("nn" 1.0)))
("reval" (("pm" 1.0)))
("veval" (("nn" 1.0)))
("rival" (("nn" 1.0)))
("tival" (("nn" 0.8889) ("**" 0.1111)))
("vival" (("**" 1.0)))
("ikval" (("nn" 1.0)))
("skval" (("nn" 1.0)))
("alval" (("nn" 1.0)))
("omval" (("nn" 1.0)))
("arval" (("nn" 1.0)))
("urval" (("nn" 1.0)))
("dsval" (("nn" 1.0)))
("esval" (("nn" 1.0)))
("gsval" (("nn" 1.0)))
("ssval" (("nn" 1.0)))
("tsval" (("nn" 1.0)))
("atval" (("nn" 1.0)))
("ntval" (("nn" 1.0)))
("utval" (("nn" 1.0)))
("ivval" (("nn" 1.0)))
("nyval" (("nn" 1.0)))
("doxal" (("av" 1.0)))
("rayal" (("**" 1.0)))
("royal" (("**" 0.8667) ("pm" 0.1333)))
("afael" (("pm" 1.0)))
("chael" (("pm" 1.0)))
("ikael" (("pm" 1.0)))
("srael" (("pm" 1.0)))
("aabel" (("pm" 1.0)))
("babel" (("pm" 1.0)))
("dabel" (("av" 1.0)))
("eabel" (("av" 1.0)))
("fabel" (("nn" 1.0)))
("iabel" (("av" 1.0)))
("kabel" (("av" 0.5833) ("nn" 0.4167)))
("mabel" (("pm" 1.0)))
("nabel" (("av" 0.5) ("pm" 0.5)))
("pabel" (("av" 1.0)))
("rabel" (("av" 1.0)))
("sabel" (("av" 1.0)))
("tabel" (("av" 1.0)))
("ubbel" (("pn" 0.7857) ("nn" 0.2143)))
("äbbel" (("nn" 1.0)))
("bibel" (("nn" 0.8) ("pm" 0.2)))
("nibel" (("av" 1.0)))
("ribel" (("av" 1.0)))
("sibel" (("av" 1.0)))
("xibel" (("av" 1.0)))
("imbel" (("pm" 1.0)))
("nobel" (("pm" 0.5) ("av" 0.5)))
("sobel" (("nn" 1.0)))
("erbel" (("nn" 1.0)))
("ssbel" (("pm" 1.0)))
("jubel" (("nn" 1.0)))
("rubel" (("nn" 1.0)))
("möbel" (("nn" 1.0)))
("pöbel" (("nn" 1.0)))
("arcel" (("pm" 1.0)))
("urcel" (("pm" 1.0)))
("mycel" (("nn" 1.0)))
("radel" (("nn" 1.0)))
("sadel" (("nn" 1.0)))
("iddel" (("pm" 1.0)))
("uddel" (("nn" 1.0)))
("dedel" (("nn" 1.0)))
("iedel" (("pm" 1.0)))
("jedel" (("nn" 1.0)))
("medel" (("nn" 1.0)))
("sedel" (("nn" 1.0)))
("tedel" (("nn" 1.0)))
("eidel" (("pm" 0.5) ("nn" 0.5)))
("fidel" (("pm" 1.0)))
("akdel" (("nn" 1.0)))
("ckdel" (("nn" 1.0)))
("åkdel" (("nn" 1.0)))
("lldel" (("ab" 1.0)))
("andel" (("nn" 0.954) ("pm" 0.046)))
("endel" (("pm" 0.8148) ("nn" 0.1852)))
("indel" (("nn" 1.0)))
("ondel" (("nn" 1.0)))
("ändel" (("pm" 1.0)))
("ardel" (("pm" 1.0)))
("erdel" (("nn" 1.0)))
("ärdel" (("nn" 1.0)))
("ördel" (("nn" 1.0)))
("dsdel" (("nn" 1.0)))
("esdel" (("nn" 1.0)))
("psdel" (("nn" 1.0)))
("tsdel" (("nn" 1.0)))
("audel" (("pm" 1.0)))
("cheel" (("pm" 1.0)))
("steel" (("**" 1.0)))
("iefel" (("NT" 1.0)))
("tefel" (("nn" 1.0)))
("affel" (("nn" 1.0)))
("yffel" (("nn" 1.0)))
("ckfel" (("nn" 1.0)))
("alfel" (("nn" 1.0)))
("arfel" (("nn" 1.0)))

SIX LETTER ENDING LEXICON (EXCERPT)

("eresan" (("nn" 1.0)))
("gresan" (("nn" 1.0)))
("iresan" (("nn" 1.0)))
("mresan" (("nn" 1.0)))
("nresan" (("nn" 1.0)))
("rresan" (("nn" 1.0)))
("sresan" (("nn" 1.0)))
("tresan" (("nn" 1.0)))
("vresan" (("nn" 1.0)))
("rtisan" (("nn" 0.5) ("**" 0.5)))
("kvisan" (("nn" 1.0)))
("svisan" (("nn" 1.0)))
("tvisan" (("nn" 1.0)))
("hälsan" (("nn" 1.0)))
("remsan" (("nn" 1.0)))
("mensan" (("in" 1.0)))
("uensan" (("nn" 1.0)))
("rdosan" (("nn" 1.0)))
("sdosan" (("nn" 1.0)))
("lmosan" (("nn" 1.0)))
("prosan" (("nn" 1.0)))
("oppsan" (("in" 1.0)))
("farsan" (("nn" 0.6667) ("pm" 0.3333)))
("hassan" (("pm" 1.0)))
("kassan" (("nn" 1.0)))
("massan" (("nn" 1.0)))
("sessan" (("nn" 0.9444) ("pm" 0.0556)))
("dissan" (("nn" 1.0)))
("mossan" (("nn" 1.0)))
("jässan" (("nn" 1.0)))
("mässan" (("nn" 0.7931) ("pm" 0.2069)))
("bössan" (("nn" 1.0)))
("mössan" (("nn" 1.0)))
("ljusan" (("av" 1.0)))
("1gatan" (("pm" 1.0)))
("agatan" (("pm" 1.0)))
("dgatan" (("nn" 0.75) ("pm" 0.25)))
("egatan" (("pm" 0.9524) ("nn" 0.0476)))
("ggatan" (("pm" 0.8) ("nn" 0.2)))
("lgatan" (("pm" 1.0)))
("ngatan" (("pm" 0.8333) ("nn" 0.1667)))
("ogatan" (("pm" 1.0)))
("rgatan" (("pm" 1.0)))
("sgatan" (("pm" 0.8511) ("nn" 0.1489)))
("tgatan" (("pm" 1.0)))
("ugatan" (("pm" 1.0)))
("ygatan" (("pm" 1.0)))
("ögatan" (("pm" 1.0)))
("skatan" (("nn" 1.0)))
("rlatan" (("nn" 1.0)))
("dbetan" (("nn" 1.0)))
("uletan" (("nn" 1.0)))
("tiftan" (("nn" 1.0)))
("lyftan" (("nn" 1.0)))
("ängtan" (("nn" 1.0)))
("olitan" (("pm" 1.0)))
("rlitan" (("nn" 1.0)))
("kritan" (("nn" 1.0)))
("uritan" (("nn" 1.0)))
("isitan" (("nn" 1.0)))
("gvitan" (("nn" 1.0)))
("vaktan" (("nn" 1.0)))
("ruktan" (("nn" 1.0)))
("lyktan" (("pm" 0.5) ("nn" 0.5)))
("mältan" (("nn" 1.0)))
("lantan" (("nn" 1.0)))
("tentan" (("nn" 1.0)))
("pontan" (("av" 1.0)))
("juntan" (("nn" 1.0)))
("luntan" (("nn" 1.0)))
("muntan" (("nn" 1.0)))
("jäntan" (("nn" 1.0)))
("läntan" (("pm" 1.0)))
("räntan" (("nn" 1.0)))
("väntan" (("nn" 1.0)))
("skotan" (("nn" 1.0)))
("kartan" (("nn" 0.7143) ("pm" 0.2857)))
("jortan" (("nn" 1.0)))
("järtan" (("nn" 1.0)))
("märtan" (("nn" 1.0)))
("värtan" (("pm" 0.6667) ("nn" 0.3333)))
("sastan" (("pm" 1.0)))
("iestan" (("nn" 1.0)))
("restan" (("pm" 1.0)))
("kistan" (("pm" 0.7442) ("nn" 0.2558)))
("listan" (("nn" 1.0)))
("nistan" (("pm" 1.0)))
("rkstan" (("nn" 1.0)))
("dostan" (("nn" 1.0)))
("erstan" (("nn" 1.0)))
("orstan" (("nn" 1.0)))
("dsstan" (("pm" 1.0)))
("lustan" (("nn" 1.0)))
("nystan" (("nn" 1.0)))
("nästan" (("ab" 0.9977) ("nn" 0.0023)))
("västan" (("nn" 1.0)))
("röstan" (("nn" 1.0)))
("gattan" (("nn" 1.0)))
("hattan" (("pm" 1.0)))
("lattan" (("nn" 1.0)))
("mattan" (("nn" 1.0)))
("hettan" (("nn" 1.0)))
("mittan" (("nn" 1.0)))
("kottan" (("nn" 1.0)))
("lottan" (("nn" 0.9756) ("pm" 0.0244)))
("rottan" (("nn" 0.75) ("pm" 0.25)))
("hyttan" (("pm" 1.0)))
("nyttan" (("nn" 1.0)))
("hättan" (("pm" 1.0)))
("pättan" (("nn" 1.0)))
("råttan" (("nn" 1.0)))
("skutan" (("nn" 0.6667) ("pm" 0.3333)))
("alutan" (("nn" 1.0)))
("snutan" (("nn" 1.0)))
("-rutan" (("nn" 1.0)))
("drutan" (("nn" 1.0)))
("grutan" (("nn" 1.0)))
("mrutan" (("pm" 1.0)))
("prutan" (("nn" 1.0)))
("srutan" (("nn" 1.0)))
("örutan" (("pp" 0.75) ("ab" 0.25)))
("ldytan" (("nn" 1.0)))
("rdytan" (("nn" 1.0)))
("ffytan" (("nn" 1.0)))
("ukytan" (("nn" 1.0)))
("enytan" (("nn" 1.0)))
("gsytan" (("nn" 1.0)))
("nsytan" (("nn" 1.0)))
("psytan" (("nn" 1.0)))
("vsytan" (("nn" 1.0)))
("ktytan" (("nn" 1.0)))
("otytan" (("nn" 1.0)))
("tuytan" (("nn" 1.0)))
("trätan" (("nn" 1.0)))
("dgåtan" (("nn" 1.0)))

("rtauan" (("nn" 1.0)))
("ascuan" (("nn" 1.0)))
("kaduan" (("nn" 1.0)))
("-tsuan" (("pm" 1.0)))
("chavan" (("pm" 1.0)))
("aravan" (("nn" 0.6667) ("pm" 0.3333)))
("skivan" (("nn" 0.9722) ("pm" 0.0278)))
("llivan" (("pm" 1.0)))
("halvan" (("nn" 0.75) ("pm" 0.25)))
("salvan" (("nn" 1.0)))
("hemvan" (("av" 1.0)))
("onovan" (("pm" 1.0)))
("dsovan" (("av" 1.0)))
("larvan" (("nn" 1.0)))
("kurvan" (("nn" 1.0)))
("härvan" (("nn" 1.0)))
("ldsvan" (("av" 1.0)))
("ngsvan" (("nn" 1.0)))
("dluvan" (("pm" 1.0)))
("druvan" (("pm" 1.0)))
("gruvan" (("nn" 1.0)))
("trävan" (("nn" 1.0)))
("ngåvan" (("pm" 1.0)))
("sgåvan" (("nn" 1.0)))
("tgåvan" (("nn" 1.0)))
("schwan" (("**" 1.0)))
("kläxan" (("nn" 1.0)))
("vläxan" (("nn" 1.0)))
("playan" (("nn" 1.0)))
("bunyan" (("pm" 1.0)))
("aroyan" (("pm" 1.0)))
("plazan" (("nn" 1.0)))
("tarzan" (("pm" 1.0)))
("iazzan" (("nn" 1.0)))
("ode1en" (("pm" 1.0)))
("rde1en" (("pm" 1.0)))
("ude1en" (("pm" 1.0)))
("ofe1en" (("pm" 1.0)))
("gre1en" (("pm" 1.0)))
("ite1en" (("nn" 1.0)))
("ssiaen" (("pm" 1.0)))
("saaben" (("pm" 1.0)))
("staben" (("nn" 1.0)))
("rabben" (("nn" 1.0)))
("jobben" (("nn" 1.0)))
("nobben" (("nn" 1.0)))
("gubben" (("nn" 1.0)))
("lubben" (("nn" 0.8298) ("pm" 0.1702)))
("tubben" (("nn" 1.0)))
("lleben" (("nn" 1.0)))
("ängben" (("nn" 1.0)))
("leiben" (("**" 1.0)))
("ickben" (("nn" 1.0)))
("rokben" (("nn" 1.0)))
("äskben" (("nn" 1.0)))
("kolben" (("**" 1.0)))
("bomben" (("nn" 0.9474) ("pm" 0.0526)))
("fenben" (("nn" 1.0)))
("ännben" (("nn" 1.0)))
("eroben" (("nn" 1.0)))
("verben" (("nn" 1.0)))
("rdsben" (("nn" 1.0)))
("olsben" (("nn" 1.0)))
("önsben" (("nn" 1.0)))
("-puben" (("nn" 1.0)))
("revben" (("nn" 1.0)))
("byxben" (("nn" 1.0)))
("träben" (("nn" 1.0)))
("adåben" (("nn" 1.0)))
("ie2cen" (("nn" 1.0)))
("mmacen" (("nn" 1.0)))
("gracen" (("nn" 1.0)))
("tracen" (("nn" 1.0)))
("rvicen" (("nn" 1.0)))
("rancen" (("nn" 1.0)))
("e1scen" (("nn" 1.0)))
("adscen" (("nn" 1.0)))
("ljscen" (("nn" 1.0)))
("alscen" (("nn" 1.0)))
("doscen" (("nn" 1.0)))
("ioscen" (("nn" 1.0)))
("erscen" (("nn" 1.0)))
("örscen" (("nn" 1.0)))
("dsscen" (("nn" 1.0)))
("gsscen" (("nn" 1.0)))
("ksscen" (("nn" 1.0)))
("nsscen" (("nn" 1.0)))
("tsscen" (("nn" 1.0)))
("ttscen" (("nn" 1.0)))
("utscen" (("nn" 1.0)))
("entcen" (("pm" 1.0)))
("-baden" (("pm" 1.0)))
("dbaden" (("nn" 1.0)))
("lbaden" (("nn" 1.0)))
("sbaden" (("pm" 1.0)))
("öbaden" (("pm" 1.0)))
("igaden" (("nn" 1.0)))
("piaden" (("nn" 1.0)))
("ckaden" (("nn" 1.0)))
("ikaden" (("nn" 1.0)))
("lkaden" (("nn" 1.0)))
("bladen" (("nn" 1.0)))
("kladen" (("nn" 1.0)))
("lladen" (("nn" 1.0)))
("emaden" (("nn" 1.0)))
("anaden" (("pm" 1.0)))
("dnaden" (("nn" 1.0)))
("enaden" (("nn" 1.0)))
("gnaden" (("nn" 1.0)))
("knaden" (("nn" 1.0)))
("lnaden" (("nn" 1.0)))
("onaden" (("nn" 1.0)))
("pnaden" (("nn" 1.0)))
("tnaden" (("nn" 1.0)))
("vnaden" (("nn" 1.0)))
("ånaden" (("nn" 1.0)))
("spaden" (("nn" 1.0)))
("araden" (("nn" 1.0)))
("braden" (("pm" 1.0)))
("draden" (("nn" 1.0)))
("eraden" (("pm" 0.5) ("nn" 0.5)))
("graden" (("nn" 1.0)))
("kraden" (("nn" 1.0)))
("lraden" (("nn" 1.0)))
("nraden" (("nn" 1.0)))
("traden" (("nn" 1.0)))
("äraden" (("nn" 1.0)))
("asaden" (("nn" 1.0)))
("ssaden" (("nn" 1.0)))
("staden" (("nn" 0.9805) ("pm" 0.0195)))
("avaden" (("nn" 1.0)))
("redden" (("nn" 1.0)))
("vidden" (("nn" 1.0)))
("rodden" (("pm" 1.0)))
("gudden" (("pm" 1.0)))
("kudden" (("nn" 1.0)))
("bädden" (("nn" 1.0)))
("rädden" (("nn" 1.0)))

SEVEN LETTER ENDING LEXICON (EXCERPT)

("khausen" (("pm" 1.0)))
("shausen" (("pm" 1.0)))
("opausen" (("nn" 1.0)))
("gahusen" (("nn" 1.0)))
("rahusen" (("nn" 1.0)))
("tahusen" (("pm" 1.0)))
("adhusen" (("nn" 1.0)))
("dehusen" (("nn" 1.0)))
("öghusen" (("nn" 1.0)))
("mihusen" (("nn" 1.0)))
("ukhusen" (("nn" 1.0)))
("onhusen" (("nn" 1.0)))
("rnhusen" (("nn" 1.0)))
("arhusen" (("pm" 1.0)))
("erhusen" (("nn" 1.0)))
("ashusen" (("nn" 1.0)))
("dshusen" (("nn" 1.0)))
("eshusen" (("nn" 1.0)))
("gshusen" (("nn" 1.0)))
("kshusen" (("nn" 1.0)))
("othusen" (("pm" 1.0)))
("sthusen" (("nn" 1.0)))
("euhusen" (("pm" 1.0)))
("ruhusen" (("nn" 1.0)))
("ivhusen" (("nn" 1.0)))
("rähusen" (("nn" 1.0)))
("göhusen" (("nn" 1.0)))
("måhusen" (("nn" 1.0)))
("eliusen" (("pm" 1.0)))
("-ljusen" (("nn" 1.0)))
("kljusen" (("nn" 1.0)))
("lljusen" (("nn" 1.0)))
("oljusen" (("nn" 1.0)))
("rljusen" (("nn" 1.0)))
("-blusen" (("nn" 1.0)))
("bbmusen" (("nn" 1.0)))
("armusen" (("nn" 1.0)))
("tfrusen" (("av" 1.0)))
("ratusen" (("nl" 1.0)))
("retusen" (("nl" 1.0)))
("ontusen" (("nl" 1.0)))
("iotusen" (("nl" 1.0)))
("lotusen" (("nn" 1.0)))
("lvtusen" (("nl" 1.0)))
("våtusen" (("nl" 1.0)))
("klavsen" (("pm" 1.0)))
("stavsen" (("pm" 1.0)))
("nalysen" (("nn" 1.0)))
("eljäsen" (("pm" 1.0)))
("-pjäsen" (("nn" 1.0)))
("epjäsen" (("nn" 1.0)))
("lpjäsen" (("nn" 1.0)))
("npjäsen" (("nn" 1.0)))
("spjäsen" (("nn" 1.0)))
("tpjäsen" (("nn" 1.0)))
("ogräsen" (("nn" 1.0)))
("deväsen" (("nn" 1.0)))
("seväsen" (("nn" 1.0)))
("tiväsen" (("nn" 1.0)))
("llväsen" (("nn" 1.0)))
("erväsen" (("nn" 1.0)))
("ksväsen" (("nn" 1.0)))
("nsväsen" (("nn" 1.0)))
("ssväsen" (("nn" 1.0)))
("tsväsen" (("nn" 1.0)))
("ktväsen" (("nn" 1.0)))
("ntväsen" (("nn" 1.0)))
("rtväsen" (("nn" 1.0)))
("inlösen" (("nn" 1.0)))
("ansösen" (("nn" 1.0)))
("ndtösen" (("nn" 1.0)))
("gsbåsen" (("nn" 1.0)))
("andåsen" (("pm" 1.0)))
("eleåsen" (("pm" 1.0)))
("takåsen" (("nn" 1.0)))
("senåsen" (("pm" 1.0)))
("alpåsen" (("nn" 1.0)))
("llpåsen" (("nn" 1.0)))
("ttpåsen" (("nn" 1.0)))
("teråsen" (("pm" 1.0)))
("tesåsen" (("nn" 1.0)))
("ogsåsen" (("nn" 1.0)))
("ensåsen" (("pm" 0.5) ("nn" 0.5)))
("abbaten" (("nn" 1.0)))
("didaten" (("nn" 1.0)))
("oldaten" (("nn" 1.0)))
("ekfaten" (("nn" 1.0)))
("legaten" (("nn" 1.0)))
("regaten" (("nn" 1.0)))
("ariaten" (("nn" 1.0)))
("asiaten" (("pm" 1.0)))
("ntiaten" (("nn" 1.0)))
("dikaten" (("nn" 1.0)))
("vokaten" (("nn" 1.0)))
("relaten" (("nn" 1.0)))
("ramaten" (("pm" 1.0)))
("rnmaten" (("nn" 1.0)))
("lomaten" (("nn" 1.0)))
("tomaten" (("nn" 1.0)))
("ismaten" (("nn" 1.0)))
("tumaten" (("nn" 1.0)))
("penaten" (("nn" 1.0)))
("senaten" (("nn" 1.0)))
("ionaten" (("nn" 1.0)))
("sonaten" (("pm" 1.0)))
("paraten" (("nn" 0.9722) ("pm" 0.0278)))
("adraten" (("nn" 1.0)))
("feraten" (("nn" 1.0)))
("piraten" (("pm" 0.75) ("nn" 0.25)))
("okraten" (("nn" 0.7143) ("pm" 0.2857)))
("åkraten" (("nn" 1.0)))
("amraten" (("nn" 1.0)))
("toraten" (("nn" 1.0)))
("citaten" (("nn" 1.0)))
("aktaten" (("nn" 1.0)))
("ultaten" (("nn" 1.0)))
("antaten" (("nn" 1.0)))
("dstaten" (("nn" 1.0)))
("estaten" (("nn" 1.0)))
("istaten" (("nn" 1.0)))
("lstaten" (("nn" 1.0)))
("sstaten" (("nn" 1.0)))
("tstaten" (("nn" 1.0)))
("ervaten" (("nn" 1.0)))
("ämbeten" (("nn" 1.0)))
("arbeten" (("nn" 1.0)))
("årbeten" (("nn" 1.0)))
("rofeten" (("nn" 1.0)))
("udgeten" (("nn" 1.0)))
("bbheten" (("nn" 1.0)))
("ddheten" (("nn" 1.0)))
("gdheten" (("nn" 1.0)))
("ldheten" (("nn" 1.0)))
("mdheten" (("nn" 1.0)))
("ndheten" (("nn" 1.0)))

("odheten" (("nn" 1.0)))
("rdheten" (("nn" 1.0)))
("agheten" (("nn" 1.0)))
("ggheten" (("nn" 1.0)))
("igheten" (("nn" 0.9981) ("pm" 0.0019)))
("ugheten" (("nn" 1.0)))
("riheten" (("nn" 1.0)))
("ckheten" (("nn" 1.0)))
("ikheten" (("nn" 1.0)))
("okheten" (("nn" 1.0)))
("skheten" (("nn" 1.0)))
("elheten" (("nn" 1.0)))
("llheten" (("nn" 1.0)))
("ålheten" (("nn" 1.0)))
("amheten" (("nn" 0.9879) ("pm" 0.0121)))
("omheten" (("nn" 1.0)))
("umheten" (("nn" 1.0)))
("ymheten" (("nn" 1.0)))
("ömheten" (("nn" 1.0)))
("enheten" (("nn" 1.0)))
("mnheten" (("nn" 1.0)))
("nnheten" (("nn" 1.0)))
("änheten" (("nn" 1.0)))
("önheten" (("nn" 1.0)))
("roheten" (("nn" 1.0)))
("ppheten" (("nn" 1.0)))
("arheten" (("nn" 1.0)))
("erheten" (("nn" 1.0)))
("ärheten" (("nn" 1.0)))
("isheten" (("nn" 1.0)))
("ssheten" (("nn" 1.0)))
("ösheten" (("nn" 1.0)))
("ktheten" (("nn" 1.0)))
("ltheten" (("nn" 1.0)))
("ntheten" (("nn" 1.0)))
("stheten" (("nn" 1.0)))
("ttheten" (("nn" 1.0)))
("ätheten" (("nn" 1.0)))
("ovheten" (("nn" 1.0)))
("rvheten" (("nn" 1.0)))
("yvheten" (("nn" 1.0)))
("nyheten" (("nn" 1.0)))
("råheten" (("nn" 1.0)))
("vdieten" (("nn" 1.0)))
("paketen" (("nn" 1.0)))
("raketen" (("nn" 1.0)))
("gsmeten" (("nn" 1.0)))
("maneten" (("pm" 1.0)))
("rpoeten" (("nn" 1.0)))
("vpoeten" (("nn" 1.0)))
("tapeten" (("nn" 1.0)))
("bareten" (("nn" 1.0)))
("citeten" (("nn" 1.0)))
("diteten" (("nn" 1.0)))
("liteten" (("nn" 1.0)))
("miteten" (("nn" 1.0)))
("niteten" (("nn" 1.0)))
("riteten" (("nn" 1.0)))
("siteten" (("nn" 1.0)))
("titeten" (("nn" 1.0)))
("uiteten" (("nn" 1.0)))
("viteten" (("nn" 1.0)))
("ulteten" (("nn" 1.0)))
("erteten" (("nn" 1.0)))
("esteten" (("nn" 1.0)))
("edveten" (("av" 1.0)))
("amveten" (("nn" 1.0)))
("enveten" (("av" 1.0)))
("odaften" (("**" 1.0)))
("skaften" (("nn" 1.0)))
("kraften" (("nn" 1.0)))
("pgiften" (("nn" 1.0)))
("rgiften" (("nn" 1.0)))
("vgiften" (("nn" 1.0)))
("skiften" (("nn" 1.0)))
("driften" (("nn" 0.913) ("pm" 0.087)))
("kriften" (("nn" 0.9306) ("pm" 0.0694)))
("stiften" (("nn" 1.0)))
("hälften" (("nn" 1.0)))
("ldoften" (("nn" 1.0)))
("mdoften" (("nn" 1.0)))
("rtoften" (("nn" 1.0)))
("rluften" (("nn" 1.0)))
("sluften" (("nn" 1.0)))
("tluften" (("nn" 1.0)))
("dhäften" (("nn" 1.0)))
("shäften" (("nn" 1.0)))
("llöften" (("nn" 1.0)))
("slöften" (("nn" 1.0)))
("tlöften" (("nn" 1.0)))
("lechten" (("**" 1.0)))
("hichten" (("**" 1.0)))
("sichten" (("**" 1.0)))
("habiten" (("nn" 1.0)))
("elbiten" (("nn" 1.0)))
("inbiten" (("av" 1.0)))
("nnbiten" (("nn" 1.0)))
("erbiten" (("av" 1.0)))
("lsbiten" (("nn" 1.0)))
("pediten" (("nn" 1.0)))
("rediten" (("nn" 1.0)))
("anditen" (("nn" 1.0)))
("treiten" (("**" 1.0)))
("ulfiten" (("nn" 1.0)))
("rofiten" (("nn" 1.0)))
("aeliten" (("nn" 1.0)))
("meliten" (("nn" 1.0)))
("seliten" (("nn" 1.0)))
("elliten" (("nn" 1.0)))
("dsliten" (("vb" 1.0)))
("gsliten" (("av" 1.0)))
("rsliten" (("vb" 1.0)))
("tsliten" (("vb" 1.0)))
("nyliten" (("nn" 1.0)))
("limiten" (("nn" 1.0)))
("raniten" (("nn" 1.0)))
("eoriten" (("nn" 1.0)))
("loriten" (("nn" 1.0)))
("voriten" (("nn" 1.0)))
("spriten" (("nn" 1.0)))
("visiten" (("nn" 1.0)))
("aptiten" (("nn" 1.0)))
("estiten" (("nn" 1.0)))
("inviten" (("nn" 1.0)))
("-sviten" (("nn" 1.0)))
("asviten" (("nn" 1.0)))
("dsviten" (("nn" 1.0)))
("esviten" (("nn" 1.0)))
("msviten" (("nn" 1.0)))
("nsviten" (("nn" 1.0)))
("tsviten" (("nn" 1.0)))
("ysviten" (("nn" 1.0)))
("flöjten" (("pm" 1.0)))
("idakten" (("nn" 1.0)))
("ndakten" (("nn" 1.0)))
("chakten" (("nn" 1.0)))
("gjakten" (("nn" 1.0)))
("sjakten" (("nn" 1.0)))

APPENDIX F - TEXT FILES

These texts are the ones used in the SWETWOL run.

FILE NAME	REFERENCE
ADOP	Madeleine Kats *Adoptivbarn växer upp* Bonnier Fakta Bokförlag AB, Stockholm 1990
ANNA	Anna Bergenström *Annas mat – stora kokboken* Bonnier Fakta Bokförlag AB, Stockholm 1991
ALPERNA	Eva Hellström-Boström *Vandra i alperna* Bonnier Fakta Bokförlag AB, Stockholm 1991
DIREKTIV	Jusdep (Genre HA), Statsrådsberedningen *Författningar och utredningsdirektiv* Stockholm 1990
DJUR	Åke Aronson and Peter Eriksson *Djurens spår* Bonnier Fakta Bokförlag AB, Stockholm 1990
DN	(Newspaper articles) *Dagens Nyheter* Stockholm 1990
DONAU	Claudio Magris *Donau* Bokförlaget Forum AB, Stockholm 1990
GALA	Tim McGirk *Gala* Bokförlaget Forum AB, Stockholm 1990
LOVSONG	Stephen Vizinczey *Lovsång till den mogna kvinnan* Bokförlaget Forum AB, Stockholm 1990
MATTOR	Knut Larson *Orientaliska mattor* Bonnier Fakta Bokförlag AB, Stockholm 1990
OPERA	Göran Järvefelt *Opera Regi – ett sökande efter människan* Bonnier Fakta Bokförlag, Stockholm 1990
PARADIS	Virginia Andrews *Paradisets portar* Bokförlaget Forum AB, Stockholm 1990
UNT	(Newspaper articles) *Upsala Nya Tidning* Uppsala 1990

APPENDIX G - PROPOSED TAGS TO EXAMPLES

Here the module's tagging of all examples listed in the paper is given. Source text files are not specified. The examples are ordered according to the paragraphs in which they appear. Swedish letters *å*, *ä*, *ö* have been substituted for SWETWOL representation *}*, *{*, *|*.

§ 2

```
("*bourbon-*dampierre" (("pm" 1.0)))
```

§ 2.1

```
("*casteel" (("**" 1.0)))
("*luke" (("av" 0.8571) ("pm" 0.1429)))
("*allavaara" (("pm" 1.0)))
("*jokkmokkstrakten" (("nn" 1.0)))
("*arabi-belutch" (("nn" 0.75) ("pm" 0.1875) ("**" 0.0625)))
("*azerbaijandistriktet" (("nn" 1.0)))
("möbeloch" (("pm" 1.0)))
("*bronsoijoch" (("kn" 0.9996) ("pm" 0.0003) ("NT" 0.0001)))
("*fayet/*st" (("ab" 0.6004) ("nn" 0.2017) ("av" 0.1042) ("pm" 0.0343)
  ("vb" 0.0257) ("kn" 0.0137) ("**" 0.0111) ("an" 0.0067) ("in" 0.001)
  ("pp" 0.001) ("NT" 0.0003)))
```

§ 2.2

```
("t.o.m." (("an" 1.0)))
("cm2" (("an" 1.0)))
("km2" (("an" 1.0)))
("*a*c:s" (("an" 1.0)))
("enl" (("an" 1.0)))
("*w*e*a" (("av" 0.3179) ("vb" 0.2494) ("nn" 0.1618) ("pn" 0.1527)
  ("ab" 0.0522) ("pm" 0.0328) ("nl" 0.0157) ("an" 0.0096) ("**" 0.004)
  ("in" 0.0024) ("pp" 0.0008) ("kn" 0.0005) ("NT" 0.0002) ("al" 0.0)))
("*m*c*a" (("av" 0.3179) ("vb" 0.2494) ("nn" 0.1618) ("pn" 0.1527)
  ("ab" 0.0522) ("pm" 0.0328) ("nl" 0.0157) ("an" 0.0096) ("**" 0.004)
  ("in" 0.0024) ("pp" 0.0008) ("kn" 0.0005) ("NT" 0.0002) ("al" 0.0)))
("*r*a*f" (("pm" 0.3584) ("nn" 0.3102) ("an" 0.1886) ("**" 0.1224)
  ("pp" 0.0171) ("in" 0.0016) ("av" 0.0008) ("vb" 0.0008)))
("*e*f*t*a" (("av" 0.3179) ("vb" 0.2494) ("nn" 0.1618) ("pn" 0.1527)
  ("ab" 0.0522) ("pm" 0.0328) ("nl" 0.0157) ("an" 0.0096) ("**" 0.004)
  ("in" 0.0024) ("pp" 0.0008) ("kn" 0.0005) ("NT" 0.0002) ("al" 0.0)))
```

§ 2.3

```
("robinsonäventyr" (("nn" 1.0)))
("folköverflyttningar" (("nn" 1.0)))
("heratimönstrade" (("av" 0.3333) ("vb" 0.6667)))
("coverversion" (("nn" 1.0)))
("cooköarna" (("nn" 0.6739) ("pm" 0.3261)))
("dirndlkjol" (("nn" 1.0)))
("drangförfattare" (("nn" 1.0)))
("leclerc-arméns" (("nn" 0.4991) ("vb" 0.1708) ("pn" 0.1511) ("pm"
  0.1261) ("ab" 0.0458) ("**" 0.0064) ("av" 0.0003) ("NT" 0.0003)
  ("an" 0.0001)))
("fnkonferensen" (("nn" 1.0)))
```

§ 2.4

```
("*djursholms-*bromma-*lidingö-gängen" (("nn" 0.8377) ("pn" 0.1265)
  ("pm" 0.0182) ("kn" 0.011) ("vb" 0.0049) ("**" 0.0013) ("av"
  0.0005)))
("inte-jag" (("pn" 0.9942) ("nn" 0.0058)))
("sånt-är-livet-filosofi" (("nn" 1.0)))
("juan-complex" (("nn" 0.875) ("av" 0.125)))
("*m*b*d-barn" (("nn" 1.0)))
("*b*v*c-mottagning" (("nn" 1.0)))
("fot-i-fot" (("nn" 0.9667) ("pm" 0.0333)))
("bra-känsla" (("nn" 1.0)))
("*x*i*i*i-sviten" (("nn" 1.0)))
("karpatisk-balkansk-bysantiska" (("av" 1.0)))
("jagvetintevad" (("av" 1.0)))
("*dannemora/österby" (("pm" 1.0)))
("*hornstein/*voristan" (("nn" 0.6275) ("pm" 0.3725)))
("göra-få" (("vb" 0.8632) ("av" 0.1353) ("nn" 0.0014)))
("hungrig-mat-mätt" (("vb" 0.5) ("av" 0.4286) ("ab" 0.0714)))
("du-och-jag-ensamma-i-världen" (("nn" 1.0)))
("baby-på-nytt" (("av" 0.9852) ("pm" 0.0148)))
("jag-förstår-dig-inte" (("ab" 0.9999) ("**" 0.0001)))
```

§ 2.5

```
("g~l" (("pp" 0.351) ("nn" 0.306) ("vb" 0.1163) ("av" 0.0648) ("ab"
  0.0618) ("pm" 0.0537) ("an" 0.0184) ("pn" 0.0116) ("**" 0.0101)
  ("kn" 0.0057) ("in" 0.0002) ("NT" 0.0002) ("nl" 0.0001)))
("k*antsyning" (("nn" 1.0)))
("karderier" (("nn" 1.0)))
("stadkant" (("nn" 1.0)))
("stader" (("nn" 1.0)))
("numdahs" (("pm" 1.0)))
("våfflad" (("vb" 1.0)))
```

§ 2.6

```
("+" ((NONE 0.0)))
("=" ((NONE 0.0)))
("&" ((NONE 0.0)))
("???" ((NONE 0.0)))
("<<" ((NONE 0.0)))
("+" ((NONE 0.0)))
```

§ 2.7

```
("000--160" (("an" 1.0)))
("10,25x21,50" (("an" 1.0)))
("60--100x120--180" (("an" 1.0)))
("12-15-årsålder" (("nn" 1.0)))
("8-uddig" (("av" 1.0)))
("1796-1820" (("an" 1.0)))
("-68" (("an" 1.0)))
```

§ 2.8

```
("fram-" (("NT" 1.0)))
("50-" (("NT" 1.0)))
("bochara-" (("NT" 1.0)))
("aaaaahh!" ((NONE 0.0)))
```

§ 2.9

```
("lif" (("pm" 1.0)))
("af" (("pm" 0.4718) ("nn" 0.3521) ("pp" 0.1479) ("an" 0.0141) ("**"
  0.007) ("vb" 0.007)))
("öfwer" (("pm" 1.0)))
("upptäcke" (("nn" 1.0)))
("förbjude" (("nn" 1.0)))
```

§ 2.10

```
("the" (("**" 0.9082) ("pm" 0.0918)))
("au" (("**" 0.551) ("pm" 0.4184) ("an" 0.0306)))
("revoir" (("pm" 1.0)))
("again" (("pm" 0.8049) ("**" 0.1951)))
("*abbey" (("**" 1.0)))
("51st" (("ab" 0.6004) ("nn" 0.2017) ("av" 0.1042) ("pm" 0.0343) ("vb"
  0.0257) ("kn" 0.0137) ("**" 0.0111) ("an" 0.0067) ("in" 0.001) ("pp"
  0.001) ("NT" 0.0003)))
("54th" (("pm" 0.9246) ("**" 0.0714) ("an" 0.004)))
("mkhatshwa" (("pm" 1.0)))
("mnko" (("pm" 1.0)))
("srbik" (("nn" 0.7484) ("pm" 0.148) ("av" 0.0905) ("pn" 0.0078) ("**"
  0.0047) ("NT" 0.0005)))
("crkva" (("pm" 1.0)))
("vrsac" (("pm" 0.9737) ("**" 0.0263)))
```

§ 2.11

```
("klövar.*k*a*t*t" (("nn" 0.2081) ("ab" 0.1579) ("pn" 0.1429) ("vb"
  0.1072) ("ie" 0.0868) ("al" 0.0824) ("av" 0.0815) ("kn" 0.0812)
  ("pp" 0.024) ("pm" 0.0157) ("nl" 0.0043) ("an" 0.004) ("**" 0.0031)
  ("NT" 0.0007) ("in" 0.0001)))
("cr me" (("nn" 0.4961) ("**" 0.213) ("pm" 0.1273) ("ab" 0.0701) ("av"
  0.0597) ("pn" 0.0208) ("vb" 0.0078) ("an" 0.0052)))
("g~l" (("pp" 0.351) ("nn" 0.306) ("vb" 0.1163) ("av" 0.0648) ("ab"
  0.0618) ("pm" 0.0537) ("an" 0.0184) ("pn" 0.0116) ("**" 0.0101)
  ("kn" 0.0057) ("in" 0.0002) ("NT" 0.0002) ("nl" 0.0001)))
("1950.*tekke-*bochera" (("vb" 1.0)))
("garn.*erivan" (("nn" 0.6379) ("pm" 0.3621)))
("avudnsjuk" (("av" 1.0)))
("chokartat" (("av" 0.6667) ("ab" 0.3333)))
("fascinderad" (("vb" 0.75) ("av" 0.25)))
("defenitivt" (("ab" 0.8485) ("av" 0.1515)))
("oosynlig" (("av" 1.0)))
("öpppna" (("av" 0.6429) ("vb" 0.3571)))
```

§ 2.12

```
("måsten" (("nn" 1.0)))
("*c*d-ns" (("nn" 0.4991) ("vb" 0.1708) ("pn" 0.1511) ("pm" 0.1261)
  ("ab" 0.0458) ("**" 0.0064) ("av" 0.0003) ("NT" 0.0003) ("an"
  0.0001)))
("r-ljud" (("nn" 1.0)))
("somrigare" (("av" 0.9167) ("nn" 0.0833)))
("meetinget" (("**" 1.0)))
```

§ 2.13

```
(" " ((NONE 0.0)))
("" ((NONE 0.0)))
("*" ((NONE 0.0)))
("* " ((NONE 0.0)))
("bengt-*d" (("pp" 0.4036) ("nn" 0.2392) ("vb" 0.1063) ("pm" 0.0762)
  ("ab" 0.0665) ("av" 0.0494) ("pn" 0.0453) ("**" 0.0073) ("an"
  0.0057) ("NT" 0.0003) ("in" 0.0001) ("kn" 0.0001)))
("*d*d-2:1-2-4-6" (("an" 1.0)))
("fra ois" (("pm" 0.8704) ("**" 0.1111) ("nn" 0.0185)))
("140Ø" ((NONE 0.0)))
```

§ 2.14

```
au revoir
```

§ 2.15

Not taggable

INDEX

Nota Bene! Swedish alphabetc order is preserved, i.e., *å*, *ä* and *ö* are placed after *z*.