MPhil Computer Speech and Language Processing
Department of Engineering
Word Sense Disambiguation Using CIDE+

Julie Weeds
Trinity Hall

August 29, 2000

Abstract

The semi-automatic extraction of a semantic hierarchy from a machine readable dictionary is investigated. This hierarchy could potentially be used by a word sense disambiguation technique which measures semantic relatedness of two senses using a semantic hierarchy and evaluates potential sense configurations according to this measure of semantic relatedness.

Declaration

I declare that this thesis is substantially my own work. Where reference is made to the works of others the extent to which that work has been used is indicated and duly acknowledged in the text and bibliography.

This thesis does not exceed 15000 words in length including footnotes and bibliography.

signed:

Acknowledgements

I would like to thank Judita Preiss for her work and her results which made the final evaluation possible. I would also like to thank my supervisor, Ted Briscoe, and assistant supervisor, Anna Korhonen who have supervised me throughout my project. I would further like to acknowledge EPSRC, whose funding made it possible for me to undertake this project, and Cambridge University Press whose dictionary CIDE+ was used throughout the project.

List of Figures

List of Tables

Contents

1 Introduction

Arthur: "Ah. This is obviously some strange usage of the word "safe" that I hadn't previously come across"
- Douglas Adams, The Hitchhiker's Guide to The Galaxy.

1.1 Aim

The aim of this project was to semi-automatically build a hierarchy of noun senses in the machine-readable dictionary (MRD) CIDE+ (Cambridge International Dictionary of English), using information provided therein. Such a hierarchy has many potential uses, one of which is in word sense disambiguation (WSD) tasks.

1.2 Motivation

A large number of words used in English are polysemous, that is, they have more than one sense. These senses may be closely related or completely different. In the latter case, the senses are referred to as homographs of each other. For example, the word "bank", as defined in CIDE+, has three core senses (or homographs) which are distinguished by the guidewords *organisation*, *"raised ground"* and *turn*. Within a core sense there may also be a plurality of more closely related senses. For example, the *organisation* homograph of "bank" includes the senses defined as, "an organisation where people and businesses can invest or borrow money...", "a bank of something, such as blood or human organs for medical use, is a place which stores these things for later use" and "in gambling, the bank is money that belongs to the owner and can be won by players."

In many language understanding tasks it is necessary to be able to disambiguate between the different senses of a word. For example, a machine translation system translating a piece of text from English to French would need to be able to translate the *organisation* sense of bank to "la banque" and the *"raised ground"* sense to "la rive", "la berge" or "le banc" (depending on the more finely-grained sense intended). The intended sense can usually be determined in context. For example, if the sentence also includes the word "money" (e.g. "I withdrew some money from the bank'."), it is more likely, although not necessary, that the sense of bank intended is that of an organisation where people and businesses can invest or borrow money. This contextual information might be used alone or in conjunction with other word sense cues such as part-of-speech, sense frequencies, verbs' semantic selection preferences on arguments, collocations (certain words often appear together as a phrase and thereby disambiguate each other e.g. "film review") and the subject domain of the text.

Two major approaches have emerged in the attempt to solve the WSD problem. The first of these, which will not be considered further here, is the use of large corpora from which statistics are extracted. The second approach, which is considered here, is to use MRDs.

Wilks and Stevenson (1998a) have developed a system which can be used to perform WSD for words in context using the MRD LDOCE (Longman Dictionary of Contemporary English). This system attempts to combine information from multiple weak knowledge sources (as defined by Newell 1973). After pre-processing the text (which includes tokenisation, sentence splitting, part-of-speech tagging, named

entity recognition, shallow syntactic analysis and lexical look-up), the system uses six disambiguation modules, which each provide a certain level of disambiguation. The idea is that by using a machine-based learning technique to integrate the information provided by the individual modules, improved overall results may be obtained.

The first disambiguation module in the Wilks and Stevenson 1998a system is a part-of-speech filter. This removes from consideration all senses for a particular word which are not consistent with the part-of-speech assigned by the part-of-speech tagger. Ideally a partial tagger, which marks each ambiguous word with the sense or senses it considers to be correct, would be used instead of a filter as a filter may make a mistake and remove the correct sense completely from consideration by the other modules. However, part-of-speech taggers such as the Brill (1995) tagger tend to be extremely accurate (accuracy up to 98%) and can considerably reduce the number of sense configurations to be considered by other modules. For example, Slator and Wilks (1987) cite the seemingly simple sentence, "There is a huge envelope of air surrounding the earth." They claim that there are 2856 possible sense configurations when all possible senses assignable from LDOCE are considered. This drops to 990 possible sense configurations when senses incompatible with part-of-speech are removed from consideration.

The second disambiguation module is a partial tagger which uses word definition overlap to disambiguate between senses. This module is based on the idea that the definition of the intended sense of a word is likely to have words in common with definitions of other words in the sentence. A motivating example for this approach, identified by Lesk (1986), is the phrase "pine cone". In LDOCE, "pine" has two major senses and "cone" has three major senses. The correct senses have the words "evergreen" and "tree" in common in their definitions. The Wilks and Stevenson (1998a) disambiguation module optimises the overlap of all words in a single sentence at the same time by minimising an evaluation function. The overlap for a given configuration of senses is defined as the total number of times each word appears more than once in the dictionary definitions of all the senses in the configuration. In order to reduce the amount of computation required, a simulated annealing technique is used rather than computing all possible configurations of senses.

The third and fourth disambiguation module are partial taggers which use LDOCE pragmatic codes. LDOCE pragmatic codes have a two-level structure and, when given for a word sense, indicate the subject area in which that word sense is likely to appear. The third module optimises the number of pragmatic codes of the same type in the sentence. The fourth module is based on the broad context algorithm developed by Yarowsky (1992).

The fifth disambiguation module is a partial tagger which returns the set of senses which are licensed by the selectional preference information in LDOCE. The sixth and final disambiguation module in the Wilks and Stevenson 1998a system is a collocation extractor.

It is thought that a system similar to the one described above could be implemented which would disambiguate between CIDE+ senses rather than LDOCE senses.

Preiss (2000) provides a disambiguation module for use in such a CIDE+ system. This module is similar to the second disambiguation module in the Wilks and Stevenson (1998a) system insofar as that it uses definition overlap to disambiguate

between senses. However, Preiss (2000) attempts to overcome a problem identified by Wilks and Stevenson (1998a) with their module. The problem being that definitions using synonymous words rather than identical words are penalised. Further, by only considering the overlap of word forms, the intuition that it is overlap in meaning that is important is to some extent lost.

With a view to overcoming this problem, Preiss (2000) uses the idea that truly synonymous words ought to belong to the same synset (i.e. set of synonymous words) in a semantic hierarchy such as WordNet (Miller et al (1990), Fellbaum (1998)). It is also the case that, in general, in a semantic hierarchy such as WordNet, the closer two words are together in the hierarchy, the greater their similarity in meaning. In other words, distance of separation in the hierarchy provides a measure of semantic relatedness.

Accordingly, Preiss (2000) defines a distance function to calculate the distance between any two word senses in the hierarchy. She then obtains the optimal sense configuration over an entire sentence using thirteen different "experts". Four of these maximise the CIDE+ overlap (in the same way as Wilks and Stevenson (1998)), four minimise distance between senses in the semantic hierarchy, and four maximise the CIDE+ overlap of definitions where overlap of two words is a function of their separation in the semantic hierarchy. Each one of each group of four is implemented in a different way. These different implementations are simulated annealing, exhaustive search, best-so-far simulated annealing and use of a Viterbi-style algorithm. The thirteenth expert simply picks the first CIDE+ sense for each word in the sentence. The decisions made by the experts are combined to give an overall decision.

However, any such approach relies to some extent on the semantic hierarchy used. WordNet can be used but this suffers from the drawback that it is necessary to map between CIDE+ senses and WordNet senses. Even presuming an accurate mapping can be made where the same senses are defined for a word, there remains the problem that in some cases WordNet defines two senses for a word where CIDE+ defines one and in other cases WordNet defines one sense where CIDE+ defines two. This is particularly likely where senses are very closely related. In a case where CIDE+ defines two senses and WordNet defines one, there is no way to disambiguate between the CIDE+ senses using WordNet since both CIDE+ senses map to the same WordNet sense. How much this will affect the results of the disambiguation module depends on how often this happens. I took a random sample (using a pseudo random number generator based on Marsaglia 2000) of 200 noun senses in CIDE+ and attempted to find the corresponding word sense in the WordNet noun hierarchy. I was unable to find a good mapping in 68 cases, that is, only an estimated 66% of noun senses in CIDE+ have a corresponding noun sense in WordNet. This does not, of course, necessarily limit the performance of a disambiguation module using WordNet to (an estimated) 66% since the word senses not contained in WordNet may occur infrequently in dictionary definitions or in text. However, I believe that this missing 34% must limit the performance of the module to some extent.

It follows, I believe, that optimum results for a disambiguation module such as Preiss (2000) could only be achieved using a hierarchy which includes every CIDE+ word sense. In accordance with this, it was my aim to go some way towards building such a hierarchy. Due to the limited time available, the size of CIDE+ and the complex issues involved in building a semantic hierarchy, which will be discussed later, I decided early on that my work would concentrate on building a noun hierarchy, with the possibility of going on to other parts of speech if time permitted.

2 Background

2.1 Semantic Hierarchies

In its simplest form, a semantic hierarchy is a tree-like structure in which nodes represent word senses (or meanings) and the directed links between nodes represent relations between word senses. Semantic hierarchies tend to be constructed for an individual part of speech since it is generally thought that the different concepts represented by nouns, adjectives, verbs, and adverbs are too dissimilar to be organised into a single structure.

However, there are a number of different relations which may hold between words. Considering first only nouns, the most comprehensive of relations is that of hyponymy, or informally, the ISA relation. The hyponymy relation relates each word sense to its hypernym (or superordinate) and its hyponyms (or subordinates). For example, "mammal" is a hyponym of "animal" and a hypernym of "aardvark". In general, a sense of a noun has a single hypernym and any number (which may be zero) of hyponyms.

According to Fellbaum (1998), this hierarchical structure, generated by the hyponymy relation, is implicit in the prototypical lexicographic definition of a noun. Information that is common to two dictionary entries, such as mammal and animal, is not stored in both entries. This information is stored solely in the entry for animal. Types of animal, such as mammals, are then assumed to inherit this information without it being explicitly stated.

Other common relations which may exist between nouns are synonymy ("car" is the same as "automobile"), antonymy ("defeat" is the opposite of "victory") and meronymy-holonymy (a "tooth" is a part of the "mouth"). However, these relations cannot be defined for all nouns and do not lend themselves as well to defining a hierarchical structure for nouns, as does the hyponymy relation. Consequently, the backbone of a noun semantic hierarchy will generally be the hyponymy relation. For example, Fellbaum (1998) indicates that the hierarchical representation generated by the hyponymy relation provides the central organising principle for the nouns in WordNet.

Broadly speaking, we might imagine that a similar semantic hierarchy could be and should be built also for adjectives, verbs and adverbs; that is, the other parts of speech generally considered to be content words. However, the idea of a semantic hierarchy for parts of speech other than nouns is not so intuitive. Here, I will briefly consider how constructing hierarchies for adjectives and verbs could be approached, and is approached in WordNet, and some of the problems associated therewith.

Adjectives are generally divided into two major classes. These are descriptive adjectives and relational adjectives. Descriptive adjectives, e.g. "fresh" or "stale", assign a value of an attribute to a noun. However, descriptive adjectives do not intuitively organise themselves into a tree-like structure. The intuitive way of organising descriptive adjectives is into clusters of similar words which are then opposed to other clusters through the antonymy relation. This is the way they are organised in WordNet. The lack of a tree-like structure is a problem for a disambiguation module such as discussed above, since there will not be a defined distance between all adjective pairs. It is also an issue for other systems which view lexical knowledge as an inheritance system.

Relational adjectives, e.g. "medical" or "lexical", relate or associate the noun they modify with another noun. For example, lexical knowledge is knowledge related to or pertaining to the lexicon. Accordingly, the way that WordNet organises relational adjectives is to link them to the nouns they modify. This is intuitive since relational adjectives play a similar role to that of a modifying noun. Further, although this organisation is not a tree structure, it does not pose so much of a problem as with that of descriptive adjectives since, assuming a complete noun hierarchy, a path will be defined from every relational adjective to every noun and hence to every other relational adjective. In fact, this might be an advantage for a word sense disambiguation module since adjectives can then be used to disambiguate nouns and vice versa.

The relation between verbs which most resembles the hyponymy relation between nouns is that of troponymy (coined by Fellbaum and Miller (1990)). Troponymy is a type of lexical entailment in which x is said to be a troponym of y if it can be said that to x is to y in some manner. For example, walk is a troponym of move. However, Fellbaum (1998) has found that verbs cannot easily be arranged into the tree-like structures into which nouns are arranged.

2.2 Building Semantic Hierarchies from Machine Readable Dictionaries

Copestake(1990) has investigated semi-automatically building semantic hierarchies from the MRD LDOCE. Her aim was to extract lexical semantic information from the definitions of lexical entries and represent this information in a lexical knowledge base (LKB) in the form of an inheritance system.

Copestake's approach is to extract a genus term for each word sense from its definition and then build the hierarchy recursively in a top down manner. Starting with a given word sense at the top of the tree, all of the dictionary entries for which this is the genus term are found, and then the program recurses on each child found until it reaches a word sense which is never used as a genus term.

Extraction of genus terms from dictionary definitions is discussed by Vossen and Copestake (1993). As they discuss, the traditional assumption is that the genus term for a noun definition is the syntactic head of the defining noun phrase. For example, if an *adder* is defined as *a poisonous snake* (taken from CIDE+[1]) then the genus term of *adder* is *snake*. However, Vossen and Copestake identify a number of different types of definition, found in LDOCE, which need to be treated differently.

Firstly, there are definitions where the syntactic head of the defining noun phrase is a synonym of the word being defined rather than a hypernym. This is generally the case when the syntactic head is unmodified.

Secondly, they identify definitions with complex kernels, that is, where the genus term is in fact a genus phrase. Vossen and Copestake have further identified four types of complex kernels which correspond to four different relations between words. The first of these is the *type/kind* relation which can be regarded as an explicit version of the hyponymy relation. For example, bitter is defined as *a type of dark, brown beer*. It's genus term is beer.

[1] Although Vossen and Copestakes work is based around LDOCE, I will take any examples from CIDE+ in order to illustrate the validity of their points in the context of extracting a semantic hierarchy from CIDE+ definitions.

The second type of complex kernel corresponds to a *quantity/mass* relation. For example, in CIDE+, a *note* is defined as *a piece of paper money*. This is a different relation to the ordinary hyponymy one, since a mass noun has become a count noun. However, Vossen and Copestake claim that, for general lexical representation purposes, such definitions can be treated as if they were in fact of the type/kind structure. They motivate this by saying that, in the context of dictionary definitions, a piece of material will always be material and thus all the properties of material should apply.

In the context of building a semantic hierarchy for word sense disambiguation purposes, their thinking would also seem to be sound. It seems likely that the word "money" will be much more useful than "piece" in disambiguating the word "note".

The third type of complex kernel identified by Vossen and Copestake, corresponds to a *member/group* relation. They also distinguish between group nouns, such as "band," which is defined as "a group of musicians", and non-group nouns, such as "dolmen" which is defined as "a group of stones"'. They claim that in the latter case, the relator "group" is really being used to indicate a *component/whole* relation (discussed below). They argue that, in the context of dictionary definitions, if a group is defined in terms of its members then it can be taken to inherit appropriate properties from them. With regard to the inverse relation ("member of"), they argue that since the use of complex kernels of the form "an x is a member of y" is almost entirely restricted in LDOCE to human denoting definitions, it can normally be assumed that an x is a human.

The fourth type of complex kernel corresponds to the *component/whole*, or meronymy, relation. Vossen and Copestake argue that with this relation, very little can be predicted about the word being defined. For example, "albumen" may be "the white part inside an egg" but it can in no way be considered to inherit the properties of "egg". The inverse of this relation is that something can be said to be a whole made up of different parts. However, Vossen and Copestake argue that it is not generally possible to predict how the semantics of the whole relate to the semantics of the components.

Vossen and Copestake also identify that some definitions have genus terms which are coordinated. Coordination may be via a conjunction or disjunction. They distinguish between coordination where the coordinated elements are alternatives between which a choice has to be made (e.g. a *landmark* is *a building or place which...*) and coordination where the entry word is a complex WHOLE of which the elements are components (e.g. *cutlery* is *knives, forks and spoons used...*).

Copestake 1990 also discusses sense disambiguating the selected genus term. It is little use knowing that "b" is a *letter* unless it is also known which sense of *letter* is meant. In order to sense-disambiguate the genus terms, Copestake exploits two features of LDOCE. The first is that it has a limited core vocabulary, thereby restricting the possible non-leaf nodes in the hierarchy. The second is that extra semantic information is provided for certain entries in the form of box codes (e.g. "human", "animal" etc.).

2.3 CIDE+

The MRD CIDE+ is an SGML encoded database of the Cambridge University Press' 1995 Cambridge International Dictionary of English for advanced learners. Since it is a learners' dictionary, a restricted core vocabulary is used. Each entry,

or record, in CIDE+ contains at least a unique identifier (url), a key (indicating the sense of the word), one or more parts of speech and one or more word forms. Other pieces of information which may be included in an entry are a definition, a guideword (particularly if the word form has more than one sense), a code indicating whether the "word" is a single word or a multi-word unit, information about semantic class, semantic selection preferences, subject domains, grammar codes, usage, other morphological forms of the same word, HECTOR codes, links to other entries and examples of how the word might be used.

The word forms grouped together in a CIDE+ definition may be variant spellings or they may be synonyms. For example, "biscuit", "bikkie", "bickie" and "cookie" are grouped together in this way. Since these word forms were already grouped together, they remained grouped together as WordNet equivalent synsets in the semantic hierarchy.

3 Development

Development of the hierarchy was split into modules. The conceptual process and flow of information is shown in Figure 1. Each module will be discussed in turn in subsequent subsections.

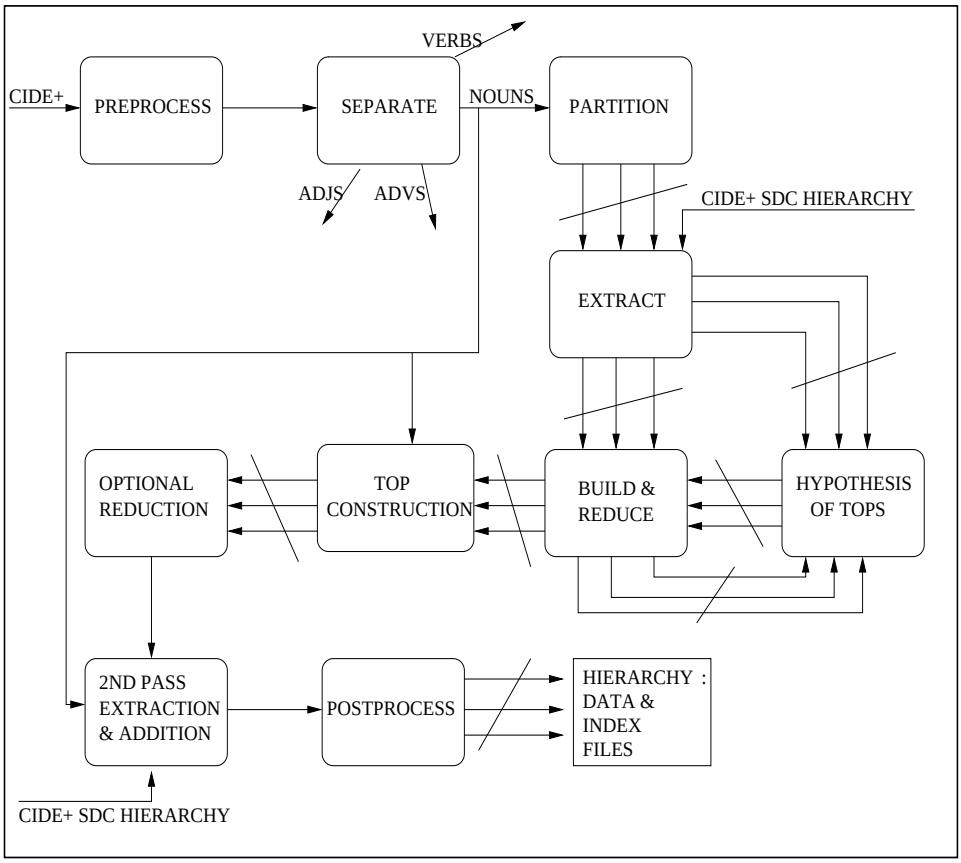

Figure 1: Development Process

Each module was written in Prolog. Prolog was chosen as the language due to its inbuilt abilities in handling large databases, recursion, searching, and list processing. Efficiency of the modules was never a primary concern due to the fact that each module should only need to be run a finite number of times during building of the hierarchy and then never again.

3.1 Preprocessing: Conversion of Database into Prolog

The first module converts the CIDE+ SGML database into a Prolog database. The CIDE+ file is read in a record at a time using code adapted from Clocksin and Mellish (1981). Each record is then converted into a set of 3-place `entry` clauses of the form *entry(Url,FieldName,FieldValue)*. The procedure that does this is such that entries for a particular field can occur any number of times and the different fields can occur in almost any order, the only requirement on order being that the `url` field occurs first. Further, certain fields, such as `eg-group` and `link` are ignored altogether as these will not be used to build the hierarchy and ignoring them reduces the memory requirements. Figure 2 shows an example of a CIDE+ input record and the corresponding Prolog clauses.

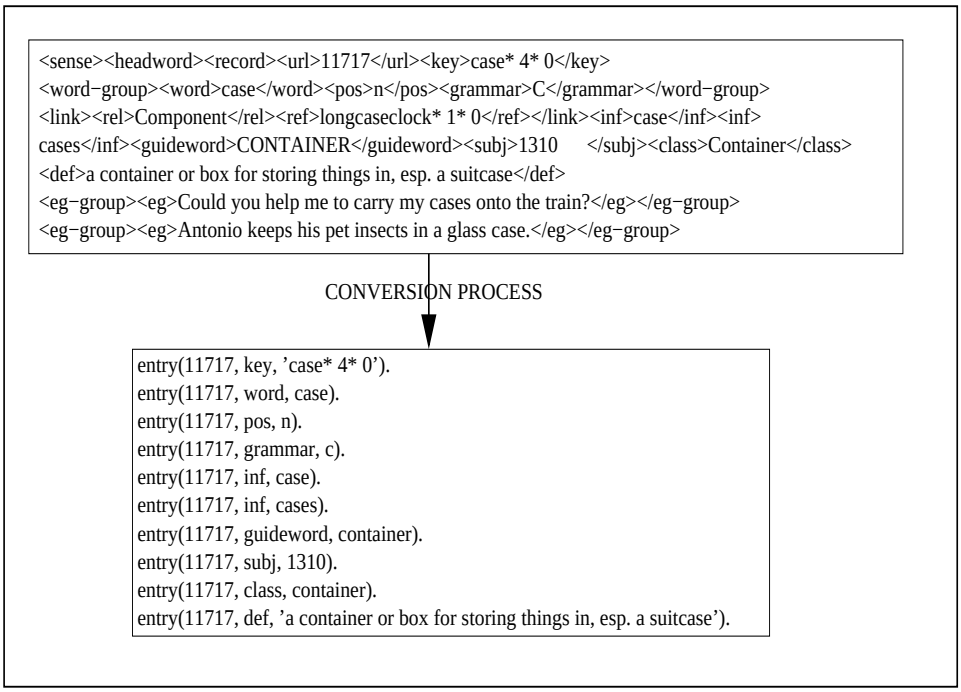

Figure 2: Conversion Process

3.2 Separation by Part of Speech

The database was separated into four smaller databases based on part of speech for two major reasons. The first was the large number of content words in the dictionary, the distribution of which is shown in Table 1. The second was the fact that, initially in any case, words of one part of speech would be linked only to other words of the same part of speech. It should be noted at this point that CIDE+ categorises content words by more than the four conventional open class parts of speech (noun, verb, adjective and adverb). For example, nouns which are usually used in their plural form, such as bacteria, are given the part of speech `pl n`. However, the separation process was into the four conventional open class parts of speech and, this being the case, plural nouns were reclassified simply as nouns.

However, a problem with separating the dictionary into separate parts of speech

Part of Speech	Number	Percentage of Total
Noun	39024	53%
Verb	14975	21%
Adjective	13834	19%
Adverb	5366	7%
Total	73199	

Table 1: Distribution of Content Words Between Parts of Speech

Part of Speech	Number of Words	Number Undefined	Percentage Undefined
Noun	39024	6676	17%
Verb	14975	2791	19%
Adjective	13834	4222	31%
Adverb	5366	3315	62%
Total	73199	17004	23%

Table 2: Distribution of Word Senses Without Definition

is that a significant proportion of word senses (approximately 23%) are not actually given a definition in CIDE+ (see Table 2).

These seemingly undefined word senses rely on the user being able to read definitions for previous related senses. These previous related senses may be of any part of speech. For example, the noun "abatement", which has CIDE+ sense key `abate* 1* 1`, is not, strictly-speaking, defined. However, the previous entry is the verb "abate", which has CIDE+ sense key `abate* 1* 0`, and this is defined as "to become less strong". This examplifies a general case where a noun has been morphologically derived from a verb.

In accordance with this feature of CIDE+, the separation module has to find a definition for each morphologically derived word. If it did not, every part of speech would have to be in memory during later stages of processing, which defeats the purpose of separating out the database in the first place.

From studying CIDE+, it would seem that the required core definition is the first one in the same major sense. This can be found relatively straightforwardly by the separation module since CIDE+ sense keys have a three part structure, that is, each key contains a word, a major sense (or homograph) number and a minor sense number. The separation module never looks for a definition outside of the same major sense on the basis that there will be a considerable difference in meanings between two such senses.

Once a definition has been found, the separation module tags it with the part of speech, the CIDE+ sense and the word that it was taken from. This is because a word such as "abatement" does not mean the same as "abate" and an appropriate definition or, at least, a genus term will have to be derived accordingly at a later stage. Table 3 shows the proportions of definitions for each part of speech which are "borrowed" from each other part of speech by the separation module. It also shows how many words are still left undefined after the separation module has been run.

The 0.02% remaining without definition corresponds to exactly 17 word senses

Part of Speech	Directly Defined	From Noun	From Verb	From Adj	From Adv	No Def
Noun	83%	4%	8%	5%	<0.5%	<0.5%(0.02%)
Verb	81%	11%	6%	2%	<0.5%	0%
Adjective	69%	18%	10%	3%	< 0.5%	<0.5%(0.04%)
Adverb	38%	20%	11%	30%	1%	<0.5%(0.03%)
Total	77%	9%	9%	6%	<0.5%	<0.5%(0.02%)

Table 3: Derivation of Definitions for Non-Core Senses

(see Appendix A for details).

The separation module also derives guidewords for non-core senses from core senses in the same way.

In general, the output of the separation module is a single clause for each entry in one of the noun_entry, verb_entry, adj_entry and adv_entry predicates (see Figure 3 for illustration). Each predicate is then stored in a separate file. However, the separation module also takes into account that some word senses are assigned more than one part of speech. For example, a single sense of the word A-frame (CIDE+ sense `a* 1* 3`) is assigned both noun and adjective part of speech. Although this goes against the intuition that a different part of speech is a different sense of a word, the separation module puts such word senses into all appropriate part of speech predicates.

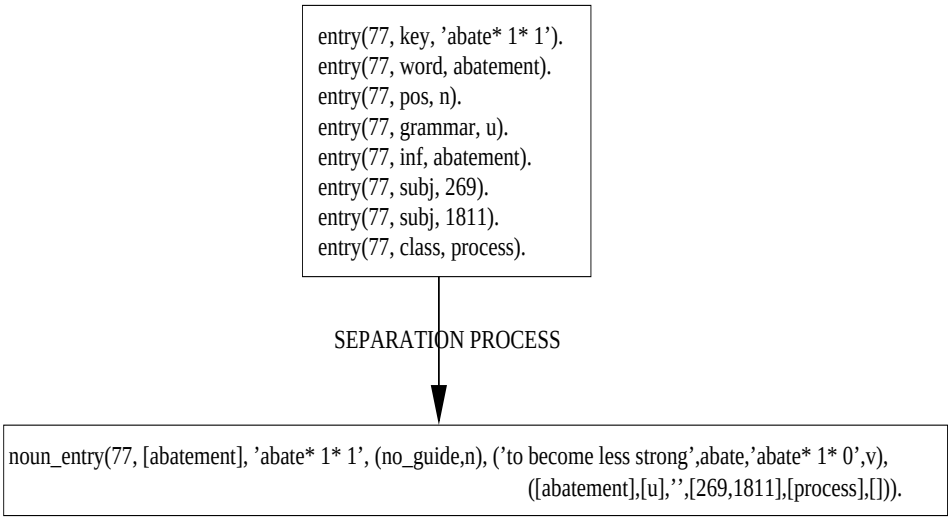

Figure 3: Separation Process

Class	Size	Class	Size	Class	Size
human	4869	activity	3751	object	3150
communication	2600	quality	2523	place	2258
event	1435	device	1339	state	1309
measurement	1242	substance	1158	group	1134
belief	1119	food	1044	abstract	1020
animal	831	body part	775	time	745
sensations	675	process	667	clothing	614
building	603	sound	533	plant	488
money	465	container	414	vehicle	406
liquid	286	drink	209	inanimate	195
energy	144	instrument	127	gas	107
solid	84	animate	36	physical	28
text	8	abstract/inanimate	1	animate/inanimate	1
belief/knowledge	1	creature	1		
no class	735				

Table 4: Distribution of Nouns Between Semantic Classes

3.3 Partitioning of Nouns into Semantic Class

CIDE+ assigns the majority of nouns to one of forty-one semantic classes. These classes have titles such as "human", "object" and "building" etc. It should be appreciated, however, that although most noun senses are assigned to a single semantic class, there are some in CIDE+ which are assigned to two or more classes and also some which are not assigned to any class. See Table 4 for the numbers of noun senses assigned to each class.

The assignment of nouns to semantic classes is primarily so that selectional restrictions or preferences can be put on verbs' arguments. For example, a verb such as "love" tends have a human as it's first argument. It follows that, in general, all of the noun senses in a semantic class should be able to be substitued for each other in a semantically-correct sentence.

Since it is generally possible to substitute a word with one of its hyponyms, it seemed likely that if a word sense was in a certain semantic class then so also would be its genus term. If a word sense was assigned to more than one semantic class then it seemed likely that its genus term would be in one or all of the same semantic classes. Alternatively, the fact that a word sense is assigned to two classes may, in some cases, be an indicator that the definition in fact contains two related senses. For example, chicken (`chicken* 1* 0`) is defined as "a type of bird..., or the meat of this bird..." and is in both animal and food classes accordingly.

Further, I also noticed that these semantic class names were similar to the set of semantic primes used to partition the nouns in WordNet (Fellbaum 1998). In WordNet, a small number of generic concepts were chosen and each one was treated as the unique beginner, or top, of a separate hierarchy.

Consequently I thought to partition the nouns according to their semantic class and then, during the genus term extraction process, only consider words in the same semantic class file as possible genus terms. Not only does this reduce the number of words to be considered at one time but it also has the advantage that it provides some level of genus term sense disambiguation. There may of course still be more

than one sense of a word in the semantic class, but the range of possibilities is reduced to ones of the same part of speech and the same semantic class. Words in two (or more) semantic classes may be found a genus term in either or both semantic classes.

However, on studying a selection of CIDE+ noun entries, I realised that the allocation of semantic classes is such that there are certain common cases where a word and its genus term are not in the same semantic class. For example, a room(`room* 1* 0`) is, I think correctly, in the *place* class. However, various types of room, such as cell (`cell* 1* 0`), are in the *building* class. Similarly, theatre (`theatre* 1* 0`) is in the *place* class but cinema (`cinema* 1* 0`), defined as a type of theatre, is in the *building* class. Another example is that various materials, (e.g. metal) are solely in the *substance* class whereas things which are pieces (e.g. ingot) or types (e.g. steel) of them, tend to be solely in the *solid* or *object* classes.

I noticed a similar problem in the *animal*, *animate* and *creature* classes. There are a large number of animals in the animal class (e.g. bird (`bird* 1* 0`)) which have the word creature as the genus term. However, the only two senses of the word creature are in the *animate* class. The existence of a *creature* class, containing a single noun the "abominable snowman", seems only to serve to confuse the matter further.

There is a further issue which seems mainly to affect the *animate* class. The *animate* class contains various words which are fairly abstract in that they do not have an obvious genus term. For example, organism and hybrid obviously define animate concepts but, being fairly abstract, they are defined by disjunctive example.[2] For example, `hybrid* 1* 0` is defined as "a plant or animal...". However, a hybrid is not "a plant or animal" in the same way that a chicken is "a bird or meat" and consequently it would be wrong to say that hybrid was a hyponym of both plant and animal. Yet, these are the only words which are available in the definition to be genus terms. Consequently, I decided that it would be necessary to consider words such as hybrid with both plant and animal classes. The issue of finding a term which encompasses both plant and animal, and is therefore truly a hypernym of hybrid, will be discussed later.

As a consequence of all these issues, I decided to combine certain classes where there might be a significant degree of overlap. It also seemed pointless to create separate trees for certain very small classes, some only containing a single noun (e.g. *animate/inanimate*) and these were also combined with one or more appropriate classes. Table 5 summarises which classes were combined with which other classes.

3.4 Primary Extraction of Genus Term

The extraction module attempts to extract a genus term from each definition. The extraction module is run once for each class file and each definition is considered in turn. Special consideration is given to definitions which must be derived from those for words of other parts of speech and the treatment of such definitions will be discussed separately.

Conceptually, there are five stages in the extraction process.

[2] I use the term disjunctive example to describe definitions which Vossen and Copestake 1993 described as having co-ordinated elements which were alternatives between which a choice has to be made.

New Class	Consisting of Old Classes
activity	activity, belief, belief/knowledge
liquid	liquid, drink
animal	animal, creature, animate
human	human, animate
plant	plant, animate
object	object, physical, substance, solid
time	time, event
abstract	abstract, abstract/inanimate
group	group, animate/inanimate

Table 5: Combination of Classes

The **first stage** is simply that the definition is tokenised into words.

The **second stage** is that an occurrence of the word being defined in the definition is replaced by a special character string (an asterisk). This is trivial, and therefore also pointless, when the word being defined is a single word. However, it is useful when the word being defined is a multi-word unit. This is because the definition is going to be considered, in future stages, a word at a time. The pre-processing of the list of words therefore allows the word being defined to always be identified and considered as a single unit. Replacing the word by a special character string also means that future stages will not attempt to tag, disambiguate or consider this word as the genus term of the definition. This disregard is justified since a word sense cannot be its own genus term as this would immediately lead to cycles in the "tree" structure. It is possible that the same word form as the word being defined could be used in the definition in a different sense (and would therefore be a valid genus term). However, this type of definition construction is very rare in CIDE+ (I have not found any examples) and is not considered.

The **third stage** is that each word is tagged as being a definition word, a class word or an other word. Definition words include words like `type`, `amount` etc. and they also include commonly occurring non-nouns such as `a`, `which`, `is` and `for`. A definition word tag includes both the fact that a word is a definition word and its type (e.g. "determiner", "equals", "component/whole"). There are three reasons for definition word tagging. The first is that these words will not be considered as possible genus terms. The second is that these words can be used to identify the location of the genus term. The third is that, by tagging the definition words with a type, it is not necessary to consider each word as a special case. For example "which", "who", "that" etc. tend to play exactly the same role in a definition, that is they tend to come after the genus term and introduce modifiers thereof. Most definition words are defined as being such over the whole of the noun part of speech. However, there are a few words which are treated differently in different classes. For example, the pronoun "someone" is generally treated as a definition word of nondescript nature. However, in the human class, it is defined as meaning "human" and this meaning is available as a `classword`. This is to cater for the extremely large number of human class definitions of the form "someone who...".

Class words are words that are also in the semantic class file under extraction. The two main issues in identifying classwords are morphological variants (i.e plurals in the case of nouns) and disambiguation between senses. Since disambiguation between senses occurs at this stage, a class word tag includes the fact that it is a class word and the appropriate CIDE+ sense key.

Plurals are simply dealt with using the *infs* field in CIDE+. The *infs* field gives other forms of the same word. However, for multi-word units it also gives the separate words of the unit and their various forms. Accordingly, it is also necessary to check the *mwu* field to ensure that a word which is part of a multi-word unit is not classified as a class word. Since the effect of a multi-word unit not being marked as such can be quite devastating to the genus term extraction process, I ran a small program to check and correct the *mwu* field of every entry before running the extraction program. I found approximately 50 multi-word units with an empty *mwu* field and was thus able to avoid the rather worrying situation where a noticeable number of the human class was determined as having the genus term `god* 1* 4` ("those who the gods would destroy, they first make mad"). It should also be noted that at this stage restricting class words to single word units will not reduce coverage any further since the definition is being considered a word at a time, and therefore, a multi-word unit cannot correctly be considered a class word.

Due to the polysemous nature of many words, even within a single semantic class, it is often necessary to choose between a number of candidate senses for a class word. The algorithm I use to disambiguate between senses scores each potential sense in the context of the word being defined. It then chooses the best sense if it is better than the next best sense by a certain threshold. If the threshold is not exceeded then the sense which occurs first in the dictionary is selected. This is based on the fact that, although CIDE+ makes no guarentees, the first sense is usually the most frequent sense and the most frequent sense is usually the correct sense. The fact that the most frequent sense is the most likely sense is especially true in a restricted vocabulary situation and considering that senses which are not of the correct part of speech and semantic class have already been eliminated.

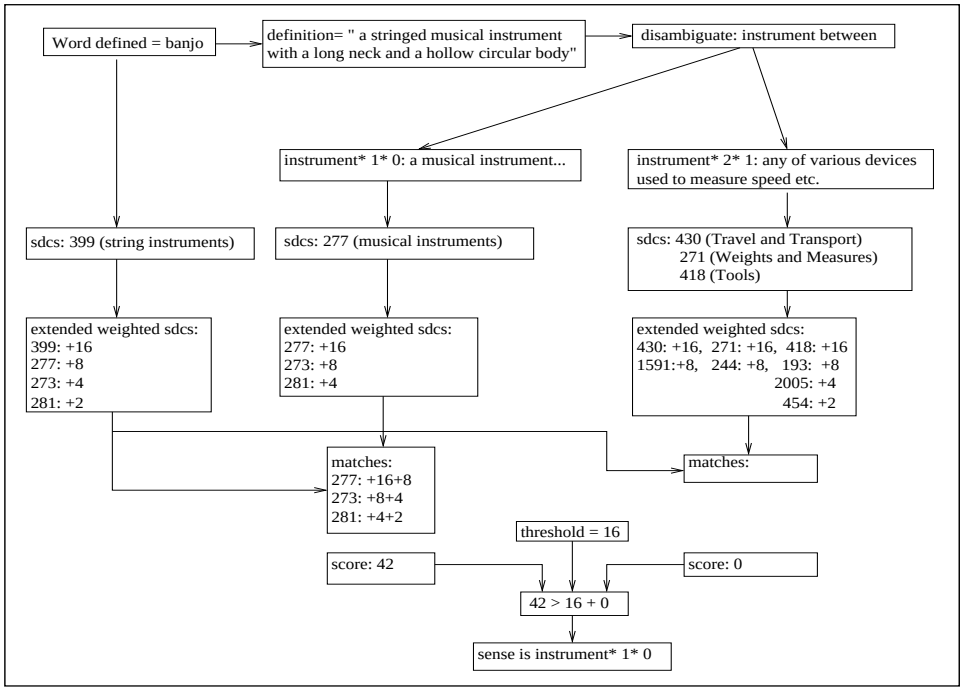

Figure 4: Disambiguation of "instrument" in Definition of "banjo"

Scoring of the potential senses relies mainly on subject domain information provided in CIDE+. However, simply matching the subject domain codes of the

word being identified with those of the sense being scored, will not achieve very much due to the large number of subject domain codes in CIDE+. CIDE+ subject domain codes have a highly hierarchical nature and, consequently, all of a subject domain code's ancestors are also considered (in a weighted manner). Figure 4 illustrates how disambiguation might be performed for a particular example.

Disambiguation also takes into account semantic class. This is more relevant when two or more classes have been combined but also relevant to words (or senses) which have dual class status. However, semantic class is given a low weighting since it is considered that most reliable information which can be derived from this has already been derived by partitioning.

The final type of word that a word in the definition may be tagged as is an "other" word. This simply applies to all words in the definition which are neither definition words or class words.

The **fourth stage** in the extraction process is to pass the tagged definition through a finite state transducer (FST) to extract a list of class words as genus term.

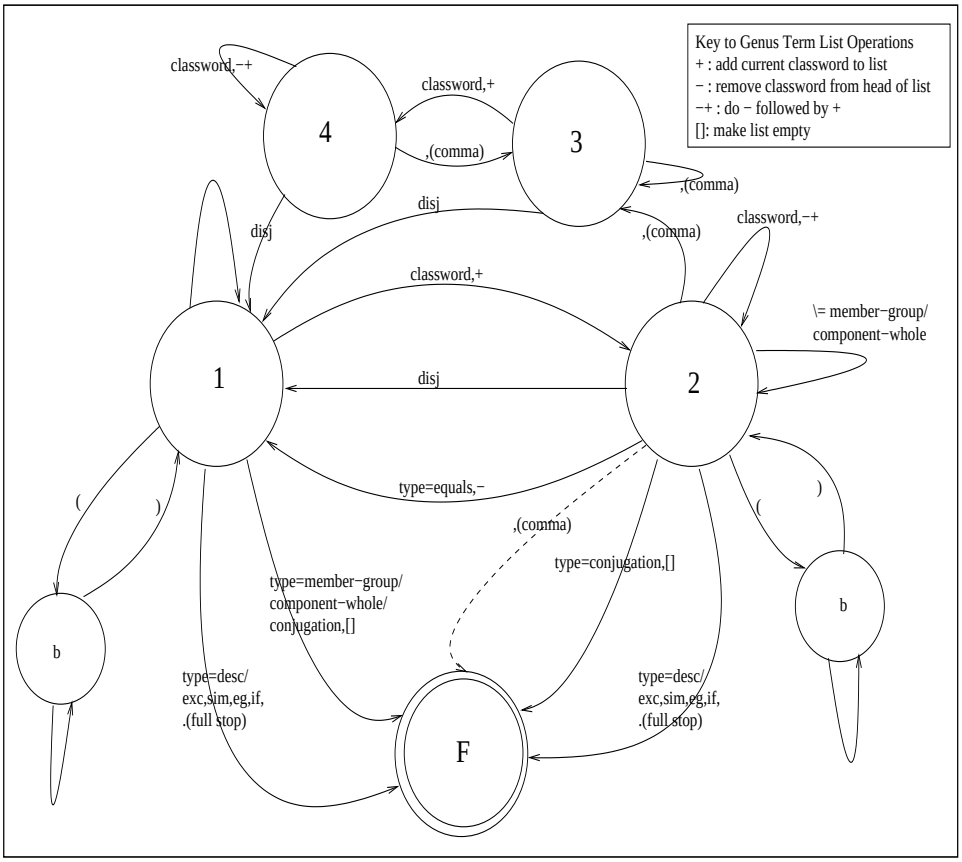

Figure 5: Simplified Extraction Finite State Transducer

The FST, a simplified version of which is shown in Figure 5, is based on Vossen and Copestake's (1993) study of dictionary definitions and also my own analysis of CIDE+ definitions. The version shown in Figure 5 covers a subset of the types of definition actually handled by the FST, and is intended for illustration purposes

only. For those interested, the actual Prolog implementation of the FST can be found in Appendix B.

It should be noted that although the transducer is largely deterministic, it is not completely so. In Figure 5, for example, there are two comma transitions out of state 2. The first is to state 3 which is a component of the subroutine which handles disjunctions. However, if this subroutine fails, i.e. a disjunction is not subsequently found, the second comma transition from state 2 to state F will be taken instead. Another example, not shown, is that the word "is" can play two different roles in a definition ("a * is a genus_term which..." or "* is informal/br/am/slang for genus_term") and its role is determined by the subsequent word(s). Both of these examples could, of course, be implemented in a completely deterministic FST, but I chose not to implement them deterministically since Prolog back-tracking handles non-determinism elegantly.

In general, the list returned by the FST will contain a single genus term. However, the list returned by the FST will be empty when the definition contains no class words or is of a certain form, e.g. "a part of a....". Further, the list may contain more than one genus term when the definition contains a disjunction. Reduction of multiple genus terms to a single genus term occurs later, during the tree-building stage, when it is possible to identify a hypernym of the disjuncts.

Although stages three and four are conceptually distinct, they are in fact implemented in tandem; that is a word is tagged and passed to the FST and then another word is tagged and so on. This is because the FST may make its decision and exit long before reaching the end of the definition. Therefore it is inefficient to perform the relatively computationally expensive operation of tagging for every word in the definition irrespective of whether or not it will be used.

The **fifth stage** of the extraction process is simply to remove any occurrence of the word sense itself from its list of genus terms.

Figure 6 illustrates the execution of the entire extraction process for a simple example.

When the genus term list returned is empty, I considered assigning the top of the class as genus term, since, although the top of the class may not be the correct direct ancestor of the word concerned, it should occur somewhere in its hypernym chain. By assigning the top of the class as genus term, the coverage of genus term extraction and consequently the tree would be significantly increased.

However, I decided against this for three main reasons. The first being that for many classes it is not possible to pick a word which can be considered to be the sole top of the class. An obvious example of such a class is the *abstract* class. However, further to such obvious examples, as will be seen later, I found during the building phase that it was necessary to use more than one top in the majority of classes. The second reason was that for dual class words, it could not be done until after extraction for all classes and then, if a genus term had still not been extracted, it would be necessary to choose a top of one of the classes. The third, and possibly most important, reason is that it is not possible to completely rely on the semantic classes assigned by CIDE+. There are two aspects of this unreliability. The first is that assigning words to a set of semantic classes is a somewhat subjective process and therefore a limited amount of variation in classes assigned by different people can only be expected. For example, is something, such as a "stone", an *object* or a *solid*? The second aspect is that a noticeable number of CIDE+ classes are indisputably questionable. To give just three examples, `midget`, defined as "a very

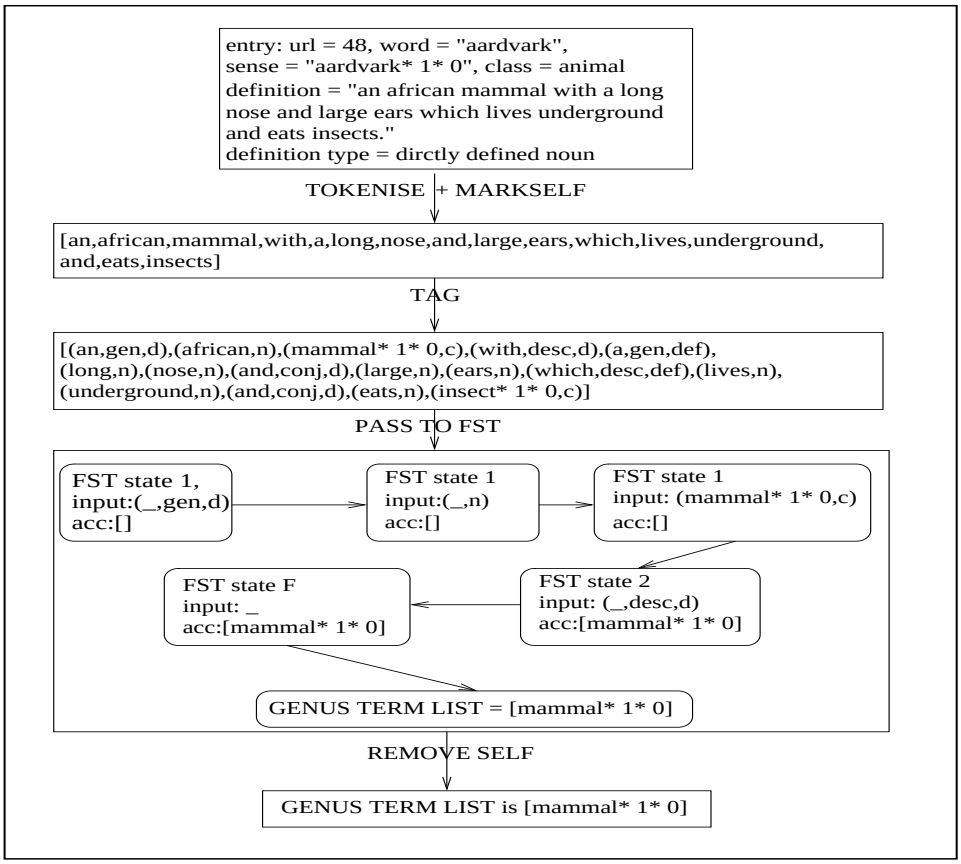

Figure 6: Example Execution of Extraction Finite State Transducer

small person", is classified as *animal* rather than *human*; `Formula One`, defined as "a type of racing car", is classified as *measurement*; and a `player` (such as a tape or record player), defined as "a machine", is classified as *human*. Consequently I decided to maintain the accuracy of the process and forego the possible extra coverage.

At this point, I should mention that alternative approaches to genus term extraction were considered. Copestake (1990) discusses using specialist parsing techniques for dictionary definitions. In accordance with this, I considered using a combination of part of speech tagging and noun phrase chunking. However, considering the unique lexicon and grammar of CIDE+ definitions, considerable time would need to be spent constructing the necessary tools or interfacing with existing tools. Further, especially considering that the accuracy of such tools is not 100 percent, I did not think the gain achieved by doing this would be great. A word which is ambiguous between adjective and noun may be more reliably tagged using existing part of speech taggers than using my combination of semantic class tagging and FST. However, the system described herein is largely accurate, see later evaluation results and, even if using accurate part of speech tagging and NP chunking, it would still be necessary to perform much of the same work performed by this system afterwards. It should be recognised that this system tags nouns using the lexicon defined by CIDE+, tagging to the level of semantic class. It further attempts to disambiguate between senses and it attempts to extract a genus term based on the

structure of the definition.

Now I will turn my attention to the extraction of genus terms from definitions borrowed from senses of other parts of speech. For example, the definition of `administrator (administrate* 1* 5)`, borrowed from that of `administrate (administate* 1* 0)`, is "to control the operation or arrangement of something". In this example, it is necessary to derive the person who controls the operation or arrangement of something.

Accordingly, for definitions borrowed from verbs, the extraction module first attempts to find the verb infinitive. This is generally quite straightforward as the majority of verb definitions are of the form "to _ in some manner". It then attempts to derive a noun from that infinitive by applying a set of morphological rules. For example, the "+er" rule incorporates the various spelling changes to construct an agent for a verb. The extraction module constructs a selection of such putative nouns and then attempts to find each in the particular semantic class file under consideration. Accordingly, in the *human* file, the process is more likely to find a corresponding agent such as "controller" and in, say, the *activity* file, the process is more likely to find a corresponding act (e.g. amusement from amuse).

I considered a similar approach to noun definitions borrowed from adjectives and adverbs. However, deriving nouns from adjectives and adverbs is not quite so straightforward. Where a noun is derived from an adjective it tends to be that the noun is something or someone who has a certain property e.g. a conservative is someone who is conservative. However, the definition of the property does not necessarily lead to the hypernym of the noun since adjectives, as discussed earlier, do not intuitively have a hierarchical structure. To deal with such nouns it would seem necessary to either rely on the assigned semantic class, e.g. a `conservative` is a *human*, or to conduct further analysis. Due to time constraints placed on the project and the fact that noun definitions borrowed from adjectives make up only a small proportion of the nouns (5%) and those from adverbs even less (<0.5%), I decided to assign such nouns an empty genus term list for the time being.

3.5 Tree Building by Class

Once the genus terms have been extracted for a class, it is then possible to build a tree (or a set of trees) for the word senses in that class.

Given a top node, the tree-building module recurses top-down throughout the class in a manner after that described by Copestake (1990). All of the nodes which have the current node in their genus term list are found and a parent-child link asserted for each. The algorithm then recurses in a depth-first manner for each child.

Care obviously has to be taken not to introduce cycles into the tree. Lexicographers try not to define concepts in a circular manner but this cannot always be avoided and, especially where some word forms have more than one genus term, it would be possible to enter into such a circle and then continue around it indefinitely. Accordingly, before asserting a parent-child link, the tree-building module checks that the would-be child is not an ancestor of the parent. It also checks that it is not already asserted as a child, to stop the same part of the tree being built more than once.

3.6 Disjunctive Definitions and Reduction of Links to the Lowest Common Ancestor

As I have discussed already, CIDE+ makes quite frequent use of the disjunctive example type of definition. For example, a `scavenger (scavenge* 1* 1)` is given the definition, "a scavenger is a bird or animal which feeds on...." Yet, it cannot be really said that in one sense a scavenger is a bird and in another it is an animal. A scavenger is actually a type of something which includes both birds and animals. Since in CIDE+, both birds and animals are defined as being types of `creature (creature* 1* 0)`, it would seem to make sense to say that a scavenger is a creature.

The tree-building module initially builds the tree so that, in this example, `creature* 1* 0` has hyponyms including `animal* 1* 0` and `bird* 1* 0`. In turn, both of these hyponyms of `creature* 1* 0` have hyponyms including `scavenge* 1* 1`. Once the initial build for a class has completed, the tree-building module makes a top-down pass through the constructed tree attempting to reduce links so that each node in the tree has only one hypernym. For each node that is found to have more than one hypernym, the algorithm locates the lowest common ancestor of its hypernyms. It then retracts the links to the node's existing hypernyms and forms a new one to this lowest common ancestor. Accordingly, in the scavenger example, the links to `animal* 1* 0` and `bird* 1* 0` will be retracted and a link to `creature* 1* 0` asserted. This procedure is carried out top-down so that, although the node under consideration may have more than one hypernym, there should generally only be one hypernym chain for each of its hypernyms.[3] This is desirable since it simplifies identification of the lowest common ancestor.

I have also already discussed that in certain cases a disjunction is used in a definition when there really is more than one sense of the word being defined. For example, `chicken* 1* 0` is defined as being "a type of bird which... or the meat of this bird which...." Here, it would not be strictly wrong to reduce the parents to the lowest common ancestor of bird and meat (`object* 1* 0`) but a lot of semantic information would be lost in the process. Further, it is not really the case that "chicken" is a type of something which includes birds and flesh, "chicken" is sometimes a "bird" and sometimes "meat". However, such examples are fairly rare, since one would expect two separate senses of a word to be defined separately. Accordingly, I ignore this possibility when reducing within in a single semantic class, assuming that the amount of semantic information lost will be minimal since both (or all) hypernymes are in the same semantic class. However, as will be discussed later, I take into account this possibility when potential hypernyms are found in different semantic classes (such as in the chicken example).

3.7 Automatic Hypothesis of Top Nodes

The tree-building module assumes that the top node (or nodes) in a particular class is known. These nodes are generally quite intuitive and I initially determined one or several top nodes for each class simply by looking at the entries and the genus terms extracted. For example, an obvious top node for the *human* class is `human* 1* 9`.

[3] There may be more than one hypernym chain from each hypernym when more than one top node has been defined for a class. However, these hypernym chains will not have any nodes in common, since otherwise they would have already been reduced by the top-down reduction algorithm. Accordingly, the existence of multiple chains for each hypernym will not affect the determination of the hypernyms' lowest common ancestor.

However, in order to increase the coverage for each class (i.e. number of nodes in tree over number of nodes with non-empty genus term list), it was necessary to find more top nodes for each class. I did this using an automatic hypothesis module.

For a particular class, the automatic hypothesis module constructs a list of all of the word senses which do not have a hypernym link but do have a genus term extracted. Then, for each item in the list, it attempts to hypothesise what a good top node would be in a bottom-up fashion. It does this by following a hypothetical link to its genus term and then considering this entry's genus term list and so on. The module decides that it has reached a potential top if a word sense does not have a genus term or if it finds a circle of definitions. Once the module has hypothesised tops for ever item in its list, it calculates how many times each hypothesised top occurs. The module returns the hypothesised top occurring the maximum number of times and its number of occurrences. The person constructing the tree can then consider this potential top node and what effect it would have on the tree.

An example of where the hypothesis module made a large impact was in the *place* class. I had already built trees starting from top nodes `place* 1* 0, land* 1* 0` and `structure* 2* 0`. However, the hypothesis module suggested adding the top node `area* 1* 0` with an expected impact of accounting for just over 700 extra noun senses.

For implementation reasons, if a class has a particularly large number of word senses unaccounted for in the tree, the hypothesis module curtails the list it considers. It then only returns an estimate of how many nodes would be added to the tree by introducing the top it hypothesises as being best.

The hypothesis module can be run repeatedly on a single class, assuming that the tree for the hypothesised best node is built, so that multiple good top nodes can be identified. Using the hypothesis module, I increased the number of top nodes from 49 to 110. This may seem like a large number of top nodes when WordNet, for example, has only 25 unique beginners. However, I believe, the number can be justified when the extent to which there is inherent tree structure in CIDE+ classes is considered.

Of course, the repeated hypothesis of tops has to stop somewhere. It has to be accepted that not all entries, not even all of those for which a genus term has been extracted, will be included in the tree. For example, the definition of `phone* 1* 0` is "a telephone". However, the only noun sense of telephone (`telephone* 1* 0`) is defined as "(to use) a phone".

Lastly, there is no reason why this module could not be run before any tree has been built and, in this way, all of the top nodes could be located by the hypothesis module. The only reason why I did not do this, was due to the amount of time taken to run the module and that I had already determined at least one top node for every class.

3.8 Construction of Top of the Hierarchy

Having constructed trees from all of the tops, the next stage was to organise the tops themselves into some sort of semantic hierarchy. This is similar to the organisation of the unique beginners in the WordNet tops file (Fellbaum 1998).

As Fellbaum (1998) discusses, the generic concepts which occur near the top of

a single hyperthetical semantic hierarchy carry little semantic information. This lack of semantic information is reflected in dictionary definitions for such concepts which is why I did not attempt to automatically extract the top of the hierarchy. Instead, the arrangement of the top semantic concepts was carried out by hand, using the dictionary entries and my own knowledge of word meanings. After the arrangement, there were seven concepts remaining as tops of their own individual trees. These concepts were *entity, condition, event, act, abstraction, phenomenon* and *group*. The other concepts, which originally topped their own trees, were now organised beneath these seven. Occasionally, I introduced a concept, which was previously unaccounted for in any tree, into this top level hierarchy so as to group other concepts together. For example, I introduced the word "life form" (`life* 1* 36`) to group the concepts of plant and creature together. I also introduced the word "abstraction" `abstract* 1* 3` in order to group various abstract concepts together. Figure 7 illustrates how some semantic concepts are linked together underneath the concept of *entity*.

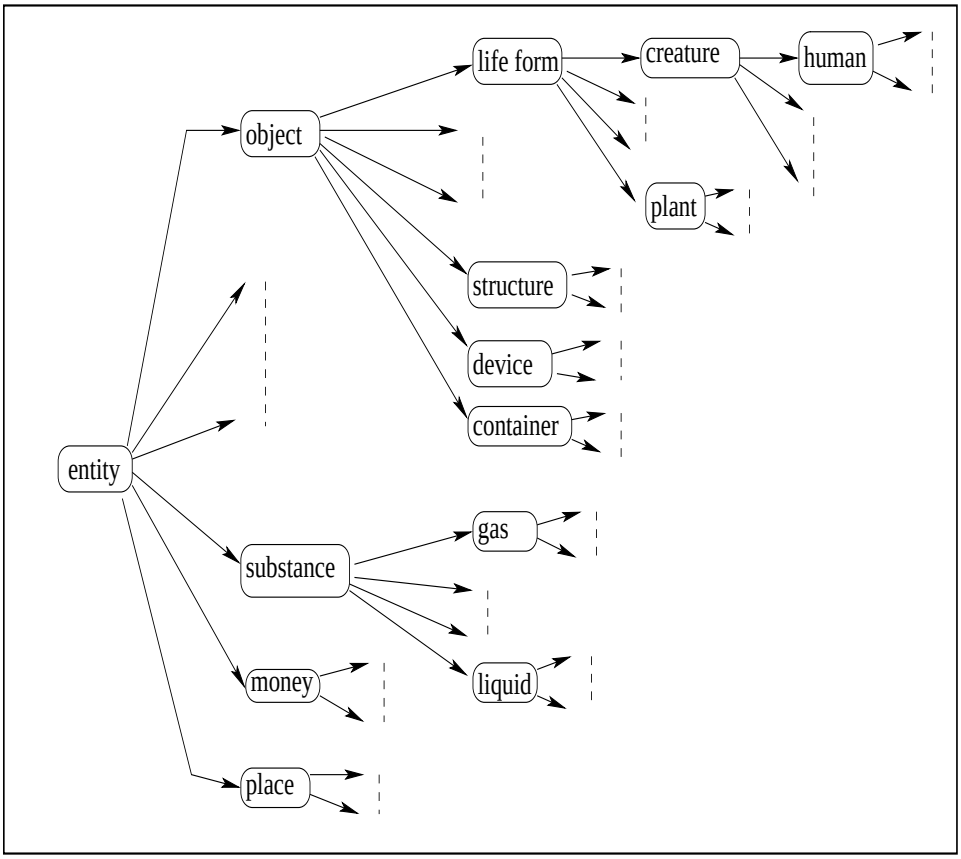

Figure 7: Hyponymic Relations Between a Subset of Top Level Concepts

3.9 Optional Reduction of Links to Lowest Common Ancestor

Once the links at the top of the semantic hierarchy had been established, I compiled the separate tree files into a single Prolog database file in which there is an entry for

each noun sense in CIDE+. In the Prolog data file each entry carries information about the word sense, such as the definition, a list of hypernym links and a list of hyponym links.

```
noun_data(1781, 'aim* 1* 2', [aim], [61101], [64337,91086],
        ('to point or direct (esp. a weapon) towards someone
        or something that you want to hit',aim,'aim* 1* 0',v)).
```

Figure 8: Sample Entry in Noun Semantic Hierarchy

Each entry in the file may represent a leaf node (i.e. it has non-empty hypernym list but empty hyponym list), or a non-leaf node (i.e. it has non-empty hypernym list and non-empty hyponym list) or, alternatively, it may not be contained in the hierarchy (i.e. it has empty hypernym list and empty hyponym list). Figure 8 is an illustration of a non-leaf node.

Ideally, in a hierarchy, each node should have only one parent node. There are two main advantages to this. The first is that a unique distance is defined between every node and every other node. The second is that the hierarchy can be used as an inheritance system without the complication of multiple inheritance. Further, a single parent node is intuitive. How can a single concept be simultaneously types of two disjoint concepts? One answer could be that it is not actually a type of either parent concept but a mixture of the two. Another could be that one concept is the parent at one time or from one perspective and the other is the parent at some other time or from some other perspective (such is the case, I believe, with the chicken example discussed previously).

Once the semantic hierarchies were compiled into a single file, it was straightforward to identify which entries had multiple hypernyms. I implemented an optional reduction module which found these entries, again in a top-down manner. The user, myself, was then given three choices. I could choose to reduce the links to the lowest common ancestor (disjunctive example definition), reduce the links to one of the parents (different word senses had been found as genus in different classes, only one of which could be considered correct), or leave the links as they were (sense correctly identified as having multiple genus terms).

3.10 Second Pass Extraction to Increase Coverage of Nouns

The second pass extraction module attempts to increase the hierarchy coverage. A slightly modified version of the original extraction module is run for each unaccounted for noun. If a genus term is found for a noun and that genus term is within the tree (i.e. it has a hypernym), appropriate links are added. If the found genus term is not within the tree a trigger is set so that subsequent addition of the genus term, triggers the addition of the appropriate links. This means that the second pass extraction can be executed in a single, sequential pass through the data file.

Modification of the original extraction module is such that any noun sense can be considered as a classword. However, since a potential genus term need no longer be from the same class file, the threshold for sense disambiguation is increased and if there is no sense which is better than other potential senses by this threshold, the word is not added. This differs from the original extraction module in that, in the original module, if the difference in scores between the best sense and other potential senses was not greater than the threshold, the first (i.e. most frequent)

sense was added. This is not done in the second pass extraction module since, without the class match, the effect of picking a wrong sense is likely to be greater.

There are also a few minor modifications to the finite state transducer itself. For example, in the original extraction process, anything in brackets in the definition is ignored on the basis that it is in brackets because it is not strictly necessary to the definition. For example, the word sense `mummy* 1* 0` has the definition, "(used by or to children) mother". However, on the second pass, words inside brackets are also considered as genus terms on the basis that a genus term was not found outside the brackets on the first pass. For example, `fresco* 1* 0` has the definition, "(a picture made by) painting on wet plaster mixture of sand, lime and water on a wall or ceiling" and is consequently, during second pass extraction, made a hyponym of `picture* 1* 0` ("a representation of someone or something produced by drawing, painting or taking a photograph").

After the second pass extraction was complete, I noticed that there were almost 2000 triggers remaining. In other words, approximately 6% of CIDE+ noun senses had had a genus term extracted but were unable to be entered in the tree since that genus term was not in the tree. Accordingly, I performed a recursive search to find the nodes that would, by their addition to the tree, cause the most other nodes to also be added to the tree. I then hand-analysed their definitions and had them added to the tree in the appropriate places. Approximately 35 word senses were hand-added in this way, triggering the addition of a further 700 nodes to the tree.

3.11 Postprocessing: Conversion into WordNet Text Format and Construction of Data and Index Files

The last stage of processing was to convert the Prolog data file into a text file (of a similar format to the WordNet `data.noun` file) and a set of index files.

This task is a fairly trivial one except in that the unique identifiers used in WordNet, and therefore likewise in these final files, are also byte offsets. In other words, a synset with the identifier 03049908 occurs at offset 03049908 in the file. The use of byte offsets allows application programs, such as Preiss 2000, to move around easily within the data file. Figure 9 illustrates the structure of the final data file (`data_noun`). It should be noted that CIDE+ word senses that are not covered by the hierarchy are not included in the final files. It should also be noted that the data file is not in alphabetical order since the second pass extraction dislocates many nouns from their original positions. The index files, illustrated in Figure 10, are, of course, ordered on the contents of their first field.

Five index files, which are required by Preiss 2000, are provided. The index file `index_count` corresponds to the WordNet file `index.noun`. This file gives a word form, the number of synsets the word form appears in and a list of those synsets in order of their estimated frequency (which is taken to be the order in which they appear in CIDE+).

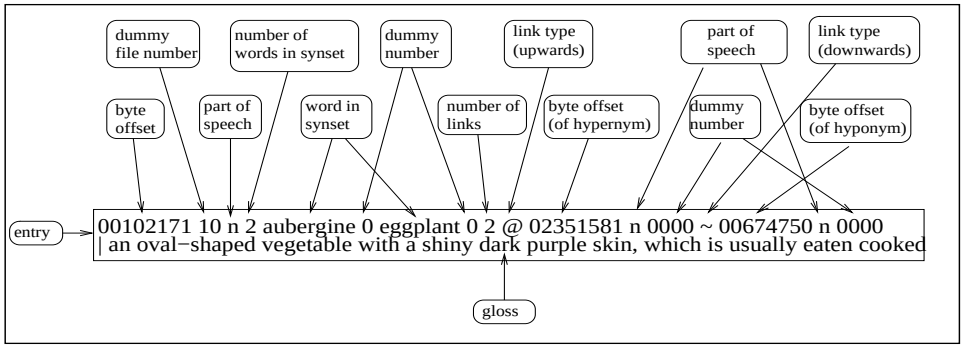

Figure 9: Annotated Entry from data_noun file

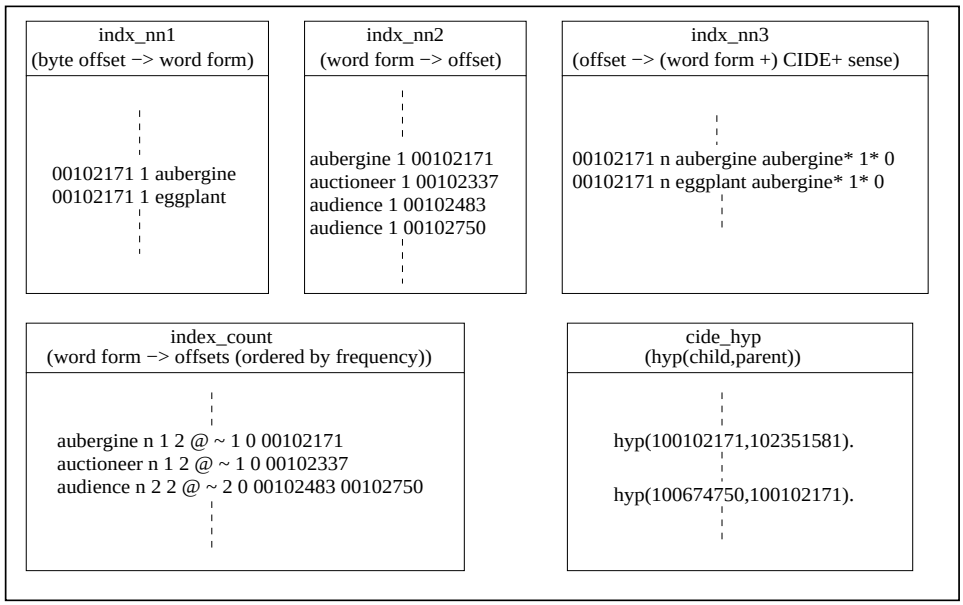

Figure 10: Illustration of the Five Index Files

Class File	Size Class(N)	Genus Terms Extracted(G)	G/N	No. of Tops(T)	No. of Nodes(L)	L/(G+T)	L/N
abstract	1020	233	23%	1	13	6%	1%
activity	4834	2634	54%	6	1545	59%	32%
animal	868	719	83%	2	709	98%	82%
body part	775	456	59%	6	177	38%	23%
clothing	614	401	65%	4	370	91%	60%
comm.	2600	1573	61%	4	729	46%	28%
container	414	307	74%	3	297	96%	72%
device	1466	962	66%	6	857	89%	58%
energy	144	85	59%	4	72	81%	50%
food	1044	743	71%	8	721	96%	69%
gas	107	84	79%	2	82	95%	77%
group	1134	818	72%	5	744	90%	66%
human	4906	3695	75%	6	3434	98%	70%
inanimate	195	10	5%	1	5	50%	1%
liquid	495	369	75%	4	359	96%	73%
meas.	1242	776	62%	2	563	72%	45%
money	465	316	68%	4	279	87%	60%
object	4414	3373	76%	4	2234	66%	51%
place	2861	2207	77%	6	1857	84%	65%
plant	524	422	81%	3	416	98%	79%
process	667	165	25%	3	91	54%	14%
quality	2523	612	24%	4	384	62%	15%
sensations	675	297	44%	4	261	86%	39%
sound	533	298	56%	4	261	86%	49%
state	1309	592	45%	4	368	62%	28%
text	8	1	13%	1	2	100%	25%
time	2180	1298	60%	5	856	66%	39%
vehicle	406	328	81%	1	305	93%	75%
noclass	735	0	0%	0	0	0%	0%
Total	39158	23774	61%	110	17973	75%	46%

Table 6: Tree Coverage By Semantic Class File

4 Evaluation

4.1 Evaluation of Coverage by Class

Table 6 shows the coverage obtained by the initial extraction process and the trees built for each class. In other words, these figures were calculated after trees had been built for all of the top nodes hypothesised by the hypothesis module but before the second pass extraction which increased coverage over the entire semantic hierarchy.

The total figures in Table 6 provide an estimate of the coverage over all the nouns in CIDE+. This estimate is slightly misleading since many nouns senses appear in more than one class. In most cases, it is only necessary for the noun sense to appear in the tree associated with one of these classes. Consequently, the actual coverage over all nouns may be higher these figures. However, these totals suggest that, at this stage, approximately 46% of CIDE+ noun senses are contained within the tree.

Class	Size of Class(N)	Genus Terms Extracted(G)	G/N	No. of Tops(T)	No. of Nodes(L)	L/(G+T)	L/N
abst.+event	10240	4789	47%	20	2806	58%	27%
act	5501	2799	51%	9	1636	58%	30%
condition	1309	592	45%	4	368	62%	28%
entity	19562	14392	74%	59	12102	84%	62%
phen.	677	383	57%	8	333	85%	49%
group	1134	818	72%	5	744	90%	66%
Total	39158	23774	61%	110	17973	75%	46%

Table 7: Tree Coverage By Generic Concept

As previously discussed, using the hypothesis of tops module, the number of tops was increased from 49 to 110. This increase in the number of tops resulted in in this estimated total coverage rising by approximately 15% (i.e. from approximately 31% to the current 46%). I believe this is a significant increase which in itself justifies the use of multiple tops per class.

The results in Table 6 also indicate that the success of the techniques used to extract genus terms and build the trees is highly dependent on the semantic class. The classes for which these techniques appear to perform best are those which fall in the generic category of "entity". This is further apparent from Table 7, which provides totals of the results in Table 6 for each of the seven generic categories (abstraction and event are combined due to the fact that some trees in the *time* class count under the abstraction concept and others count under the event concept).

Good results are also achieved for the *group* concept. However, this would appear to be statistically less significant than in the case of the *entity* concept due to the respective numbers involved.

4.2 Coverage of Entire Tree

Table 8 shows the coverage of the entire CIDE+ semantic hierarchy (CIDESH) before the second pass extraction and in its final state. It also shows WordNet coverage for a random sample of two hundred CIDE+ noun senses and the minimum and maximum expected human coverage for the same sample. I have also included an estimate of CIDESH coverage over the same sample as an indication of how representative that sample was.

The expected human coverage statistics were calculated on the assumption that the human only has access to the dictionary entries and a limited amount of inference. Minimum human coverage is what I would expect most humans to obtain accurately and would include examples like "a dog is an animal ...". To obtain maximum human coverage, a certain amount of inference would be required to deal with various types of definitions, including those which are originally for a different part of speech, those which contain disjunctive examples and those where the relation between the word being defined and the genus term is not straightforward. For example, in a case such as, "cubism is a style of modern art", there might be some disagreement between humans as to whether the genus term is "style" or "art". Examples which fall outside the range of maximum human coverage are ones which I believe it would be very difficult to extract a genus term for without other knowl-

	Coverage
CIDE+ Tree Coverage	56.4%
CIDE+ Tree Coverage before 2nd pass extraction	50.6%
Estimated WordNet Coverage	66%
Estimated Minimum Human Coverage	48.5%
Estimated Maximum Human Coverage	78%
Estimated CIDE+ Tree Coverage	61%

Table 8: Coverage Statistics

	Percentage
Hypernym correct in word and sense	55.5%
No hypernym given	39.25%
Hypernym correct in word but not in sense	1.25%
Incorrect hypernym	4%
Accuracy	94.75%

Table 9: Estimated Accuracy of Hypernyms

edge sources. For example, shortage (`short* 3* 6`) has the definition, "if there is a shortage of something, there is not enough of it."

As can be seen from the figures, the coverage of the CIDESH falls between the minimum and maximum values for human coverage, as would be expected. Evaluated on the same sample, it's coverage is 5% less than that of WordNet. The overlap of coverage between the CIDESH and WordNet is not great as one might expect. I estimated 40% of CIDE noun senses in both the WordNet hierarchy and CIDESH, 45.5% in just one of the hierarchies and 14.5% in neither hierarchy.

4.3 Accuracy of Hypernym Selection

Table 9 gives estimated figures for the accuracy of the hypernym for a given noun sense. These figures were estimated over the same sample set as used to calculate the coverage statistics. It should be appreciated that I define a noun sense to have an accurate hypernym if it does not have a wrong hypernym.

I also calculated the same set of statistics, for the same sample, before the second pass extraction was performed in order to evaluate whether the increase in coverage obtained was at the expense of accuracy. These statistics are shown in Table 10.

	Percentage
Hypernym correct in word and sense	50%
No hypernym given	46%
Hypernym correct in word but not in sense	1%
Incorrect hypernym	3%
Accuracy	96%

Table 10: Estimated Accuracy of Hypernyms Before Second Pass Extraction

As the figures show, there is a slight drop in accuracy along with the increase in coverage. However, the drop in accuracy is small (1.25%) and may even be insignificant. Further, the increase in accurate coverage (from 50% to 55.5%) would appear to be of more significance.

Leaving aside specific anomalies in CIDE+ (see Appendix A (extended version) for examples), there are a couple of main types of case where the extraction technique occasionally selects the wrong genus term or sense. The first is that occasionally the genus term is syntactically ambiguous, very often due to use of a disjunction. For example, a jar (`jar* 1* 0`) is defined as "a glass or clay container...". It is quite simple for us to see that the or coordinates the adjective constituents but it is possible that it coordinates the nbar constituents, "glass" and "clay container". Since the extraction module works on the minimal attachment principle, it will, in this example, find two genus terms, "glass" (of the drinking vessel sense) and "container". In this example, this does not affect the tree since the lowest common ancestor of "glass" and "container" is in fact the correct genus term, "container".

The second type of case is when subject domain codes reflect more heavily one part of a definition than another, and/or when an entry combines two senses. For example, "mule" (`mule* 1* 0`) is defined as "an animal whose mother is a horse and whose father is a donkey which is used for transporting loads, or fig. a person who agrees to carry illegal drugs into another country in return for payment by the person selling the drugs". Accordingly, "mule" is in both animal and human classes and the extraction module should extract "animal" when extracting the *animal* class and "person" when extracting the *human* class. However, there is also a *human* sense of animal, defined as "you can also say that a person who is very cruel or unpleasant or has no social manners is an animal". Accordingly, the extraction module extracts this as the genus term and, since the disjunction occurs much later, the rest of the definition is ignored.

4.4 Accuracy of Semantic Hierarchy Hypernym Chains

An incorrect choice in hypernym obviously has more effect on the overall hierarchy, the higher up in the hierarchy the word occurs. Accordingly, I estimated the percentage of nodes having an accurate hypernym chain to the top of the hierarchy. For a random sample set of 100 hypernym chains, I determined that 84% were completely accurate. Of the 16% inaccurate chains, 62.5% were considered to contain

	CIDESH	WordNet	LDOCE animal	LDOCE substance
average depth(est.)	5.1	6.6	-	-
maximum depth	12	>13	6	5
percentage of non-leaf nodes(P)	10.8%	-	4.2%	5%
average branching factor(100/P)	9	-	24	20
maximum branching factor	600	400	-	-

Table 11: Tree Statistics for CIDE+ Semantic Hierarchy and for WordNet

only a minor detour, that is, only a single error was made and this single error was such that the subsequent step reverted the course of the chain to the correct one.

4.5 Other Semantic Hierarchy Statistics

Table 11 summarizes certain other tree statistics, concerning depth and bushiness, for CIDESH, for WordNet (where known or estimated) and for the two hierarchies (animal and substance) constructed from LDOCE by Copestake (1990) (where known).

The average depth of CIDESH is significantly less than that of WordNet. Further, from examination of CIDESH and WordNet, there appears to be much more more variance in the length of WordNet chains. At least one chain in CIDESH reaches a length of 12 but such lengths are rare and most chains have a length of four, five or six. Connected to this is that CIDESH also appears to be bushier than WordNet. In other words, it would appear that there are a lot less non-leaf nodes in CIDESH and that each non-leaf node tends to have a lot more hyponyms. Conversely, however, the percentage of non-leaf nodes in CIDESH is higher than in Copestake's (1990) hierarchies constructed from LDOCE.

The reason that CIDESH is fairly shallow and bushy can be found in the construction of CIDE+ definitions. Being a learner's dictionary, CIDE+ defines each word sense using a limited vocabulary and it does not go into technical distinctions. Accordingly, a hypernym chain in CIDESH may be of length 7 whereas in WordNet, the same noun sense has a hypernym chain of length 14 (see Figure 11 for illustration).

4.6 Evaluation of the Semantic Hierarchy as a Tool for Word Sense Disambiguation

In order to evaluate CIDESH as a tool for WSD, CIDESH and the Preiss (2000) WSD module were tested together.

The evaluation set was randomly taken from CIDE+ examples since these are already CIDE+ sense tagged for one word in the sentence. Obviously, only sentences where the sense tag was for a noun were considered. Further, sentences containing less than two nouns, sentences where the number of senses for the sense tagged noun was less than two and sentences where one of the nouns could not be found in the appropriate hierarchy were discarded. Consequently, starting from the same initial evaluation set (of 175 sentences), CIDESH was evaluated over 78 sentences

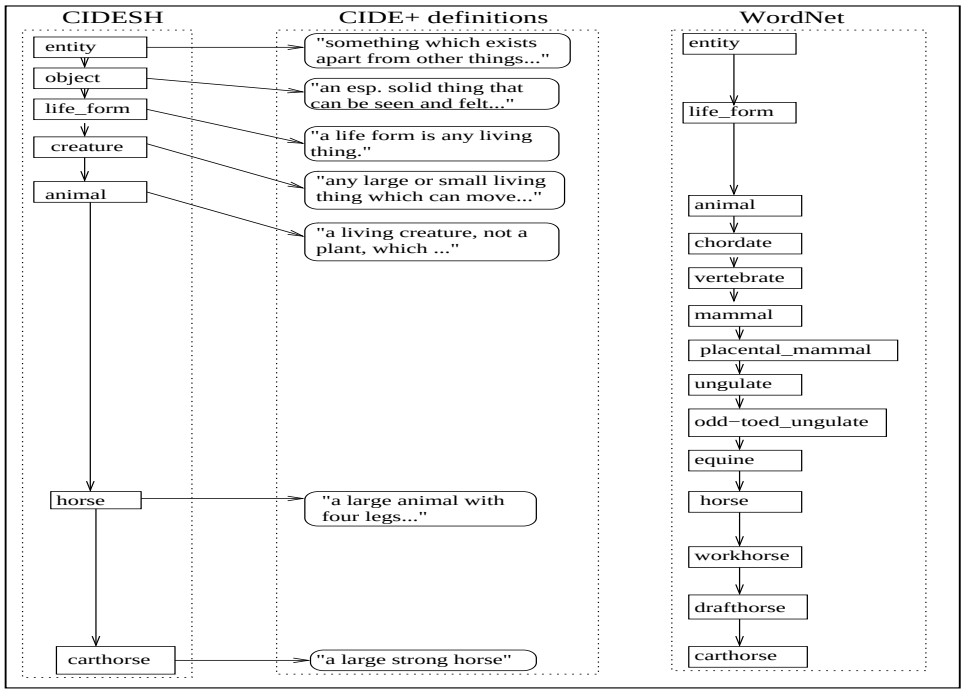

Figure 11: Comparison of CIDESH and WordNet Hypernym Chains

and WordNet was evaluated over 108 sentences.

It should be noted that the evaluation data used is not ideal for several reasons. The first is that each sentence is only sense tagged for a single word in the sentence and accordingly it is not possible to evaluate over the whole sentence. The second reason is that CIDE+ example sentences were written to illustrate the use of one particular word sense and this may mean that the other words in the sentence are unnaturally simple or unambiguous. The third reason is that no-one else has used this evaluation data and therefore the results obtained cannot easily be evaluated against the results of other people. However, these example sentences were the only available CIDE+ sense tagged data available to Preiss and myself at the time of evaluation.

Table 12 summarises the results obtained by Preiss for three of the disambiguation experts using both CIDESH and WordNet. As already discussed, Preiss implemented each of the three algorithms in four different ways. Table 12 shows just the results for the exhaustive search implementations since these achieved the best results[4].

The first of the three algorithms used by Preiss is independent of the hierarchy as is the baseline figure. These results, therefore, depend solely on CIDE+. The second algorithm used by Preiss considers just the separation in the hierarchy of the nouns in the sentence. The third algorithm considers the separation in the hierarchy of the nouns in the definitions of the nouns in the sentence.

[4]The so-called exhaustive search implementations were not strictly exhaustive since a "window" of size three was used (that is a maximum of three nouns in the sentence were considered at one time). However, this approximates quite closely a truly exhaustive search when it is considered that relatively few example sentences in CIDE+ contain more than three nouns.

Expert	Algorithm	% Accuracy using CIDESH	% Accuracy using WordNet
7	Baseline: (pick first CIDE+ sense)	29.49	31.43
3	word form overlap	46.15	50.48
4	distance in hierarchy	35.90	39.05
5	word meaning overlap	43.59	40.00

Table 12: Evaluation of CIDESH and WordNet as Tools for WSD Module

The first thing to note from the results is that using a hierarchy has not led to improved results over the standard definition overlap algorithm. However, the drop in performance using a hierarchy is not great and it is impossible to conclude whether, if the idealised hierarchy could be built, whether or not this would lead to improved results.

Secondly, considering that the definition overlap algorithm performance (and the baseline figure) is greater for the WordNet sample, a slight drop in performance could be expected for the CIDESH figures using the two hierarchy dependent algorithms without it being significant. Accordingly, I believe that for the second Preiss algorithm, CIDESH performs roughly comparably with WordNet and for the third algorithm it performs significantly better.

5 Conclusions and Further Work

The extraction and tree building techniques used to semi-automatically extract a noun hierarchy from CIDE+ have proved fairly accurate. However they have not provided the coverage of noun senses originally hoped for. Nor have they provided the coverage required in order to achieve high WSD performance, that is, even if it is possible to achieve high WSD performance using the ideas discussed herein and in Preiss (2000).

Further analysis of CIDE+ noun definitions may increase the coverage of the extraction technique, thereby increasing overall tree coverage. In particular, further analysis of definitions falling under the *abstract*, *act* and *condition* concepts may reveal why the techniques performed less well for such definitions and thus lead to a way of increasing coverage.

Using a combination of semantic class information, subject domain codes and sense frequency information seems to have proved a successful way of dealing with the sense disambiguation problem within CIDE+ definitions.

The use of the optional reduction module resulted in human interaction being required on roughly 1-2% of the entries. This compares with 5% in the case of Copestake (1990) although her criteria for entries requiring human interaction were different. She required interaction on choices about entries which might be non-leaf nodes in the hierarchy. This has the benefit that incorrect decisions can only affect a single leaf node in the hierarchy and therefore the overall accuracy of the semantic hierarchy is the same as the accuracy in hypernym selection.

A number of CIDE+ entries required human interaction because of the use of a disjunctive genus term such as "a building or place". Intuitively, buildings and places are concepts which have a lot in common. However, buildings are defined as being structures, structures as objects and objects as entities. A place, on the other hand, is a direct child of entity. This means that the lowest common ancestor of buildings and places is entity. This may be the correct hypernym for a concept defined as being "a building or place" in the sense that something can be a building or place without necessarily being either of them. However, there seems to be an enormous loss of semantic information in going from saying that a hostelry is "a bar (a place where alcoholic drinks...) or pub (a building with...)" to saying that a hostelry is "an entity".

The use of a restricted core vocabulary in CIDE+ has its advantages and disadvantages. Obvious advantages are that it makes parsing and word sense disambiguation within the definitions easier. However, it also results in a much shallower, bushier tree since the same words are used over and over again as genus term. Hence, the distribution of distances between noun senses is going to have a smaller variance and it will be more difficult to distinguish between two given noun senses.

Further, the performance of the tree-building technique suffered from the number of circular definition chains in CIDE+ and the extensive use of the disjunctive example type of definition. Both of these factors, I believe, are a result of the use of a restricted core vocabulary in CIDE+ definitions

As a tool for word sense disambiguation, the use of a custom-built hierarchy, such as CIDESH, appears to result in better performance than the use of an existing hierarchy, such as WordNet. Increased coverage in CIDESH would lead to increased coverage for the disambiguation technique and possibly also increased performance.

The lack of overlap between CIDESH and WordNet could potentially be utilised by an application and could lead to increased WSD performance. Potentially, an application could attempt to find the word sense in both hierarchies and combine the information obtained from each in some probabilistic manner.

Regarding the word sense disambiguation technique in general, I believe that this would benefit enormously from having a fully cross-linked semantic hierarchy containing all parts of speech. We do not simply rely on words of the same part of speech when trying to disambiguate a word. Even simple examples like, "I rowed/ran into the bank" require disambiguation to occur accross natural part of speech boundaries. It was noted by Fellbaum (1998) that the reason why there are so few (comparatively) verbs in the English language (and consequently why this is the most polysemous part of speech), is that most verbs rely on their noun arguments for disambiguation. Conversely, I believe that the disambiguation of many noun arguments relies on the verb of which they are an argument.

Accordingly, I believe that further work on building a complete hierarchy should concentrate on linking verbs to the nouns they take as arguments and adjectives to the nouns they modify. Verbs and adjectives of course could be further arranged amongst themselves using synonymy, tropynymy and antonymy. I believe that this is an intuitive way of arranging verbs and adjectives and would also benefit the noun hierarchy. My reason for believing this follows the argument of Keil (1979, 1983). He argues that children learn the hierarchical structure of nominal concepts by observing what can and cannot be predicated at each level. For example, the important semantic difference between inanimate and animate nouns derives from the fact that the predicates dead and alive can each be predicated for one class but not the other. Accordingly, we need the concepts "dead" and "alive" to appreciate the difference between inanimate and animate objects and this should be reflected in a noun hierarchy.

6 Bibliography

Brill, E. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. Computational Linguistics, 21(4): 543-566

Clocksin, W.F. and Mellish, C.S. 1981. Programming in Prolog. Springer-Verlag

Copestake, A. 1990. An approach to building the hierarchical element of a lexical knowledge base from a machine readable dictionary, in Proc. Of the 1st Int. Workshop on Inheritance in Natural Language Processing, Tilburg, pp. 19-29 and http://www.cl.cam.ac.uk/Research/NL/acquilex/papers.html

Fellbaum, C. 1998. WordNet: An Electronic Lexical Database. MIT Press. Includes 1993 revision of Miller et al. (1990).

Fellbaum, C. and Miller, G. 1990. Folk Psychology or Semantic Entailment? A Reply to Rips and Conrad. Psychological Review.

Keil, F.C. 1979. Semantic and Conceptual Development: An Ontological Perspective. Cambridge, Mass. Harvard University Press

Keil, F.C. 1983. On the Emergence of Semantic and Conceptual Distinctions. Journal of Experimental Psychology: General 112: 357-385

Lesk, M. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone form an ice-cream cone. In Proceedings of ACM SIGDOC Conference, pp24-26, Toronto, Canada.

Marsaglia, G. 2000. Mother-of-All Pseudo Random Number Generator. http://www.agner.org/random/mother.htm

Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D. and Miller, K. 1990. Introduction to WordNet: An On-line Lexical Database (informally known as "Five Papers on WordNet"). Journal of Lexicography, 3(4):235-244. Revised 1993, available from Princeton University. Included in Fellbaum 1998.

Newell, A. 1973. Computer models of thought and language. In Schank and Colby, editors, Artificial Intelligence and the Concept of Mind. Freeman, San Francisco, CA.

Preiss, J. 2000. Word Sense Disambiguation using CIDE+. Cambridge University MPhil in Computer Speech and Language Processing Dissertation.

Slator, B. and Wilks, Y. 1987. Towards semantic structures from dictionary entries. Technical Report MCCS-87-96, New Mexico State University

Vossen, P. and Copestake, A. 1993. Untangling definition structure into knowledge representation, in Briscoe et al. Inheritance, Defaults and the Lexicon CUP

Wilks, Y and Stevenson, M. 1998a. The Grammar of Sense. Natural Language Engineering, 4, 2, 135-144

Yarowsky, D. 1992. Word sense disambiguation using statistical models of Roget's categories trained on large corpora. ACL-Coling-92, 454-460

A Appendix A: Apparent Anomalies in CIDE+

A.1 Undefined Words

The CIDE+ entries with no obviously derivable definition are:

accreditation	-	accredited* 1* 1 (n)
blackberrying	-	blackberrying* 1* 0 (n)
disposal	-	disposal* 1* 0 (n)
interposition	-	interposition* 1* 0 (n)
reposession	-	reposession* 1* 0 (n)
resolution	-	resolution* 1* 0 (n)
seniority	-	seniority* 1* 0 (n)
statesmanship	-	statesmanship* 1* 0 (n)
subjectivity	-	subjectivity* 1* 0 (n)
accredited	-	accredited* 1* 0 (adj)
doting	-	doting* 1* 0 (adj)
feminist	-	feminist* 1* 0 (adj)
glancing	-	glancing* 1* 0 (adj)
sodding	-	sodding* 1* 0 (adj)
teeming	-	teeming* 1* 0 (adj)
sodding	-	sodding* 1* 0 (adv)
subjectively	-	subjectively* 1* 0 (adv)

A.2 Multi-Word Units with no mwu code

This is assuming that a mwu code is given according to the first `word` in a CIDE+ `word group`, which generally seems to be the case. In any case, in the majority of these entries, all words in the wordgroup are multi-word units.

sense key	word
dog* 1* 15	Every dog has it's day.
god* 1* 3	Those whom the Gods love die young.
god* 1* 4	Those who the Gods would destroy, they first make mad.
god* 2* 12	God helps those who help themselves.
pencil* 1* 8	pencil pusher
st.bernard* 1* 0	St Bernard
bonsai* 1* 8	bonsai tree
wave* 5* 1	long wave
wave* 5* 2	medium wave
wave* 5* 3	short wave
boob.tube* 1* 0	boob tube
jerrycan* 1* 0	jerry can
thermos* 1* 0	Thermos flask
doll* 1* 2	doll's pram
irish.stew* 1* 0	Irish stew
loaf* 1* 1	Half a loaf is better than none.
swiss.roll* 1* 0	Swiss roll
turkish.delight* 1* 0	Turkish delight
penny* 1* 5	In for a penny, in for a pound.
erratum* 1* 1	erratum slip
id* 1* 0	I.D. card
indian.club* 1* 0	Indian club
faith* 2* 3	Faith will move mountains.
picture* 1* 11	Every picture tells a story.
postit* 1* 0	Post-it note
interior* 1* 4	interior design
law* 1* 12	one law for the rich, another for the poor
irish.coffee* 1* 0	Irish coffee
bull* 2* 1	bull market
independent* 2* 3	Independence Day
open* 8 *7	open house
deaf* 1* 4	Those none so deaf as those who will not hear.
paranoid* 1* 3	Just because I'm paranoid doesn't mean they're not out to get me.
annual* 1* 2	annual ring
picture* 1* 3	face is a picture
search* 1* 9	search warrant
up* 1* 11	on the up and up
up* 19* 3	on the up and up
filter* 1* 1	filter bed
filter* 1* 4	filter in
tin* 1* 8	Tin Pan Alley
pandoras.box* 1* 0	Pandora's box
how* 1* 19	hows and whys
formula* 1* 2	Formula One
ironmonger* 1* 1	hardware store
register* 1* 9	registry office
b* 3* 1	B minus

B Appendix B: Selected Sections of Code

B.1 Extraction FST

```
/*****************************************************************
Genus FST for determination of genus term from noun definition
*****************************************************************/

/*-----------------------------------------------------------------
name: genus/4
arguments:
  arg1: int, +, Url of definition being analysed
  arg2: char string, +, semantic class under extraction
  arg3: list, +, list of words in definition
  arg4: list, -, list of genus terms extracted from definition
description:
  initialises 6 term predicate (gen_ext)
  which tags and implements FST
  (initialisation is: with empty genus term list
  and in state 1)
-----------------------------------------------------------------*/

genus(U,C, List, Glist) :-
        gen_ext(U,C, List, Glist,[],1).

/*-----------------------------------------------------------------
name: gen_ext/6
arguments:
  arg1: int, +, Url of definition being analysed
  arg2: char string, +, semantic class under extraction
  arg3: list, +, words remaining in definition
  arg4: list, -, list of genus terms to be returned
  arg5: list, +, accumulator - genus terms found so far
  arg6: atom, +, current state
description:
  calls extract_one which tags the next word of the definition.
  passes this to the FST genus.
-----------------------------------------------------------------*/

gen_ext(U,C,[H|T], L,A,S) :-
        extract_one(U,C,H,H1),
        genus(U,C,[H1|T], L,A,S).
gen_ext(U,C,[], L,A,S) :-
        genus(U,C,[], L,A,S).

/*-----------------------------------------------------------------
name: genus/6
arguments:
  arg1: int, +, Url of definition being analysed
  arg2: char string, +, semantic class under extraction
  arg3: list, +, remaining words in def - 1st of which is tagged
  arg4: list, -, list of genus terms to be returned
  arg5: list, +, stack accumulator -
        genus terms found so far (may be used to
        store other temporary information such as
        previous state for brackets)
  arg6: atom: +, current state
description:
```

```
   On the basis of the input (first item in the list in arg3) and
   the current state (arg6), genus makes a transition to a new
   state and may perform an operation on the stack (arg5).
   genus represents the finish state as the first state with an
   empty input. When this state is reached, the stack is unified
   with the output list of genus terms in arg4.
comments: in the case of the bracket state, the previous state is
   remembered so that it can be returned to when the matching end
   bracket is found. Accordingly this is not strictly an FST
   (where next state is dependent only on current state and input).
   However, it is equivalent to an FST where there is a separate
   bracket state associated with every main state. In this case,
   the next state would be solely dependent on current state and
   input.
-------------------------------------------------------------------*/

genus(U,C,[('(',n)|T],G,A,S) :- gen_ext(U,C,T,G,[S|A],b).
genus(U,C,[(')',n)|T],G,[S|A],b) :- gen_ext(U,C,T,G,A,S).
genus(U,C,[_|T],G,A,b) :- gen_ext(U,C,T,G,A,b).

genus(U,C,[(W,c)|T],G,A,1) :- notmem(W,A),
        gen_ext(U,C,T,G,[W|A],2).
genus(U,C,[(_,member-group,d)|_T],G,A,1) :- genus(U,C,[],G,A,1).
genus(U,C,[(_,(group,_),d)|_T],G,A,1) :- genus(U,C,[],G,A,1).
genus(U,C,[(member,_,d)|_T],G,A,1) :- genus(U,C,[],G,A,1).
genus(U,C,[(_,component-whole,d)|_T],G,A,1) :- genus(U,C,[],G,A,1).
genus(U,C,[(_,(process,_),d)|_T],G,A,1) :- genus(U,C,[],G,A,1).
genus(U,C,[(_,conj,d)|_T],G,_A,1) :- genus(U,C,[],G,[],1).
genus(U,C,[(_,equals,d)|T],G,_T2,1) :- gen_ext(U,C,T,G,[],5).
genus(U,C,[(_,equals,d)|T],G,_T2,1) :- gen_ext(U,C,T,G,[],1).
genus(U,C,[(_,(_,word),d)|T],G,A,1) :- gen_ext(U,C,T,G,A,6).

genus(_,_,[(_,member-group,d)|_T],_G,_A,2) :- !, fail.
genus(_,_,[(_,(group,_),d)|_T],_G,_A,2) :- !, fail.
genus(_,_,[(member,_,d)|_T],_G,_A,2) :- !, fail.
genus(_,_,[(_,component-whole,d)|_T],_G,_A,2) :- !, fail.
genus(_,_,[(_,(process,_),d)|_T],_G,_A,2) :- !, fail.
genus(U,C,[(_,conj,d)|_T],G,_A,2) :- genus(U,C,[],G,[],1).

genus(U,C,[(W,c)|T],G,[_H|T2],2) :- gen_ext(U,C,T,G,[W|T2],2).
genus(U,C,[(_,equals,d)|T],G,_T2,2) :- gen_ext(U,C,T,G,[],5).
genus(U,C,[(_,equals,d)|T],G,_T2,2) :- gen_ext(U,C,T,G,[],1).
genus(U,C,[(_,(_,word),d)|T],G,A,1) :- gen_ext(U,C,T,G,A,6).
genus(U,C,[(_,gen,d)|_T],G,A,2) :- genus(U,C,[],G,A,1).

genus(U,C,[(_,disj,d)|T],G,A,2) :- gen_ext(U,C,T,G,A,1).
genus(U,C,[(_,comma,d)|T],G,A,2) :- gen_ext(U,C,T,G,A,3).
genus(U,C,[(_,comma,d)|_T],G,A,2) :- genus(U,C,[],G,A,1).
genus(U,C,[(W,c)|T],G,A,3) :- notmem(W,A),
        gen_ext(U,C,T,G,[W|A],4).
genus(U,C,[(W,c)|T],G,[_H|T2],4) :- gen_ext(U,C,T,G,[W|T2],4).
genus(U,C,[(_,comma,d)|T],G,A,3) :- gen_ext(U,C,T,G,A,3).
genus(U,C,[(_,comma,d)|T],G,A,4) :- gen_ext(U,C,T,G,A,3).
genus(U,C,[(_,disj,d)|T],G,A,3) :- gen_ext(U,C,T,G,A,1).
genus(U,C,[(_,disj,d)|T],G,A,4) :- gen_ext(U,C,T,G,A,1).
genus(U,C,[(_,etc,d)|_T],G,A,3) :- genus(U,C,[],G,A,1).
genus(U,C,[(_,etc,d)|_T],G,A,4) :- genus(U,C,[],G,A,1).
genus(U,C,[(_,conj,d)|_T],G,_A,3) :- genus(U,C,[],G,[],1).
```

```
genus(U,C,[(_,conj,d)|_T],G,_A,4) :- genus(U,C,[],G,[],1).
genus(U,C,[(_,gen,d)|T],G,A,3) :- gen_ext(U,C,T,G,A,3).
genus(U,C,[(_,gen,d)|T],G,A,4) :- gen_ext(U,C,T,G,A,4).
genus(_U,_C,[(_,_,d)|_T],_G,_A,3) :- !, fail.
genus(_U,_C,[(_,_,d)|_T],_G,_A,4) :- !, fail.
genus(_,_,[],_G,_A,3) :- !, fail.
genus(_,_,[],_G,_A,4) :- !, fail.

genus(U,C,[(_,of,d)|T],G,A,1) :- gen_ext(U,C,T,G,A,1).
genus(U,C,[(_,of,d)|_T],G,A,2) :- genus(U,C,[],G,A,1).

genus(U,C,[(_,n)|T],G,A,5) :- gen_ext(U,C,T,G,A,6).
genus(_,_,[_|_T],_G,_A,5) :- !, fail.

genus(U,C,[(for,desc,d)|T],G,A,6) :- gen_ext(U,C,T,G,A,1).
genus(_,_,[_|_T],_G,_A,6) :- !, fail.

genus(U,C,[(_,prep,d)|T],G,A,1) :- gen_ext(U,C,T,G,A,1).
genus(U,C,[(_,prep,d)|_T],G,A,_) :- genus(U,C,[],G,A,1).

genus(U,C,[(_,desc,d)|_T],G,A,_) :- genus(U,C,[],G,A,1).
genus(U,C,[(_,exc,d)|_T],G,A,_) :- genus(U,C,[],G,A,1).
genus(U,C,[(_,sim,d)|_T],G,A,_) :- genus(U,C,[],G,A,1).
genus(U,C,[(_,eg,d)|_T],G,A,_) :- genus(U,C,[],G,A,1).
genus(U,C,[(_,if,d)|_T],G,A,_) :- genus(U,C,[],G,A,1).
genus(U,C,[(_,stop,d)|_T],G,A,_) :- genus(U,C,[],G,A,1).

genus(U,C,[_|T],G,A,X) :- gen_ext(U,C,T,G,A,X).

genus(_U,_C,[],nogenus,[],_).
genus(_U,_C,[],G,G,_).
```