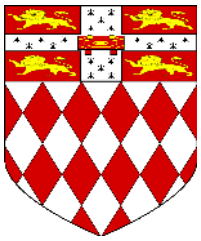
Pleonastic pronouns

by

Thomas A. Wood
Fitzwilliam College, University of Cambridge



Submitted in conformity with the requirements
for the degree of
MPhil in Computer Speech, Text and Internet Technology
(Project report)
Department of Computer Science
University of Cambridge

Abstract

Pleonastic pronouns

Thomas A. Wood

Fitzwilliam College, University of Cambridge

This report presents an investigation into the problem of identifying non-referential (pleonastic) uses of the word 'it' in English text with minimal use of tagged corpora. The basic technique involves extracting a number of grammatical and syntactic features for each instance of 'it' in a training corpus which has been anaphora-tagged and parsed. A machine learning approach is then used to produce a list of rules which can be used to identify unseen 'it's as either referential or non-referential.

The basic system gives an accuracy of 70-75% on a set of cross-genre unseen test data. Given that the baseline performance is 69%, this is not a promising result.

It is proposed that an initial heuristic of identifying all document-initial 'it's as non-referential may be as effective as a smaller, manually-tagged corpus. This procedure is implemented and tested; the results are negative.

Bootstrapping and co-training methods are then used to augment the training set with automatically tagged text. These methods also do not improve the performance above the baseline.

A performance of 89% is achieved on a limited section (41%) of the test data, indicating that a small portion of the examples are very easily classified with high confidence.

Word count: 14,999.

Chapter 1

Introduction

1.1 Anaphora resolution

In natural language text and speech, an **anaphor** is a word, such as 'it', 'he', or 'that', which refers to an object or proposition mentioned elsewhere in the discourse.[1] The identification of the noun phrase, verb phrase or other proposition to which an anaphor refers (the **antecedent** of the anaphor) is an important problem in computer natural language processing [1, 2].

The English pronoun 'it' usually acts anaphorically, that is, one or more antecedents can be identified for it. However in certain circumstances the word 'it' has no genuine antecedent, and is present only to fill a slot in the grammatical structure of a sentence. Systems which process natural language text for purposes such as anaphora resolution should be able to recognise these cases. For instance, consider the following text:

> ***It***$_1$ *kept raining.* ***The fans***$_2$ *were dismayed.* ***They***$_2$ *thought that the game was likely to end in a draw and that the series would be lost.*

A system for anaphora resolution should link the 'they' to 'the fans' but it also needs to know that the initial 'it' does not need to be resolved to anything in the text. The initial 'it' is an example of a **pleonastic** (or *expletive*) pronoun. These non-referential pronouns occur

[1] In this document, the word 'anaphor' is taken to include its traditional meanings of both 'anaphor' and 'cataphor' (anticipatory anaphor).

systematically in a variety of contexts, and exist because the English language requires verbs to have an explicit subject whether or not a semantically relevant subject exists.

Most existing anaphora resolution systems have hand-coded lists of contexts in which pronouns such as 'it' are likely to be non-referential, although there is some limited work on machine-learning approaches [3, 4]. However these rely on substantial quantities of marked up data. It would be useful to have an unsupervised algorithm which returns a binary decision as to whether a pronoun needs resolution, with minimal human intervention.

1.2 Aims of the project

My intention for this project is to investigate supervised and semi-supervised methods of identifying non-referential 'it's. I will attempt this by extracting a number of grammatical and syntactic features from a set of manually or automatically tagged examples, and using these features to train a system which is capable of classifying unseen examples as either **Y** (referential) or **N** (non-referential).

Initially I will train a classifier from manually annotated corpora. The next approach involves looking at contexts in which the 'it' is unlikely to have a genuine antecedent, such as when it is the first word in a document. I will then investigate bootstrapping techniques, whereby a trained classifier is used to augment the set of training data and this is then used to build a more advanced classifier.

1.3 Previous work on pleonastic pronouns

Theoretical analyses Huddleston and Pullum [5] and Quirk *et al.* [6] have produced grammars of the English language which give a thorough description of referential and pleonastic 'it's (referred to by Quirk *et al.* as 'placeholder "it"'). I have referred to these to produce my classification of pleonastics in Ch. 2.

Computational approaches Hobbs [7] developed an anaphora resolution algorithm which works by traversing the surface parse trees of the text from left to right, looking for noun phrases

of the correct gender and number. The algorithm is simple but works in the majority of cases. Unfortunately it ignores the possibility of pleonastic 'it's.

Lappin and Leass's [8] anaphora resolution algorithm works from salience measures derived from syntactic structure, which is generated by McCord's Slot Grammar parser [9]. Lappin and Leass report a 4% improvement on Hobbs's algorithm. Their approach does take pleonastics into account, but its performance specifically on pleonastics is not reported.

Paice and Husk [1] developed a rule-based classifier specifically for identifying pleonastic 'it's. They examined various categories of non-referential 'it's in the English language, and derived some simple surface rules which could be used for identifying them. They constructed 'templates', such as 'it **V STATUS** to **TASK**',[2] which were to be matched to sections of text surrounding instances of 'it'.

They chose the 160,000 word Section J of the Lancaster-Oslo/Bergen corpus [10] (scientific English) as their working corpus. The corpus was split into a development half and an evaluation half and the classifier was trained on the development text. When the classifier was tested, it gave an accuracy of **96.3%** on their development data and **86.5%** on the test data—although Paice and Husk later added words from the evaluation text to their decision system, improving their test set performance to 95.8% and reporting this (incorrectly) as their system performance.

Evans [4] investigated some machine learning approaches for identifying pleonastic 'it's, training and testing from the SUSANNE [11] and BNC [12] corpora.

Instances of 'it' were converted into feature vectors (containing information about the surrounding context). These were typically lemmata and part-of-speech tags of immediately preceding and following material, and information about whether the text fitted into a certain template, such as 'It was [obvious]$_{\mathbf{ADJ}}$ [the book]$_{\mathbf{NP}}$ would fall'.

A memory based learning algorithm, which used a k-nearest neighbour method to compare new feature vectors to the vectors in the training set, gave **71.5%** accuracy on an evaluation set (using a binary classification of referential vs. non-referential), whereas rule based algorithm gave **70.2%** accuracy.

Müller [13] investigated a decision rule based method of identifying pleonastic 'it's in spoken

[2] In this example, the verb **V** is a verb of being, and the **STATUS** word can be an adjective, verbal past participle, or indefinite noun phrase from a set of predetermined word lists.

multi-party dialogue. This was developed from the ICSI Meeting Corpus of meeting transcriptions [14] and involved training a decision list based classifier on a number of shallow features extracted from the transcribed text. Müller's classifier gave an accuracy of 79.6% on the test data (with recall 60.9% and precision 80.0%).

Siddharthan [2], in his Ph.D. thesis, investigated a shallow approach to syntactic text simplification. This method required a pronoun resolution module, which would obviously not need to be applied in the case of pleonastic 'it's. However, since Siddharthan worked mainly with newspaper reports, which he notes have a high proportion (around 80%) of non-referential 'it's, he did not implement a filter for detecting pleonastic occurrences of the pronoun 'it', but rather chose to treat all 'it's as pleonastic.

This unusual approach can be best understood in terms of the problem which Siddharthan was trying to solve: the purpose for his anaphora resolution system was to enable pronouns to be replaced by their antecedents, an important part of the text simplification process. The cost of a false resolution is therefore quite high, but a pronoun left unresolved will always be comprehensible to the reader.

This shows that the purpose of any pleonastic identification algorithm should be taken into account in its construction, as the relative importance of identifying a non-referential 'it' vs. a referential 'it' can often be skewed due to the risks and costs involved. In this project I will ignore these considerations, as I am concerned primarily with maximising the accuracy of a simple unbiased referential-vs.-non-referential classifier.

1.4 Corpora

I will use four corpora for the development and evaluation of the pleonastic identification system in this project. These span various genres and are summarised in Table 1.1.

The largest corpus is the **Reuters corpus** [15], which consists of a large number of articles from the Reuters news agency. This has not been annotated with pronoun references, and as such I will use it for bootstrapping experiments and investigating other methods which do not require tagged data.

The **BBN corpus** [16] consists of business news articles from the Wall Street Journal. This

Corpus	Genre	Size	Tagged	No. 'it's	No. initial 'it's
Reuters [15]	News	14,000,000 sent. (550,000 docs.)	No	1,700,000	9,000 (∼6,800 **N**)
BBN [16]	Business (WSJ)	49,207 sent. (2,312 docs.)	Yes	11,766 (9,831 **N**)	
Siddharthan Dev. [2]	Various	1,547 sent.	Yes	342 (101 **N**)	
Siddharthan Eval. [2]	Various	1,693 sent.	Yes	352 (115 **N**)	

Table 1.1: The four corpora used in the project.

corpus is tagged: all pronouns have been annotated with a number and co-indexed with any identifiable antecedents.

The Reuters and BBN texts both consist of very restricted domains (general news and business news respectively). For this reason I will also make use of a third and less domain-specific source of text, primarily for evaluation purposes: the text used by **Siddharthan** in his Ph.D. thesis [2]. This contains text from various domains: news, literature, sport, travel, comment sections from the *Guardian* newspaper [17], children's writing, and technical manuals.

I have manually assessed all of the $(342 + 352 = 694)$ occurrences of 'it' in Siddharthan's text and categorised them either as referential (**Y**), or as one of five categories of non-referential 'it's which are defined in Ch 2.3.

Since Siddharthan's text is cross-genre, it can be a reliable evaluation measure for systems developed using the Reuters and BBN corpora. The text has therefore been split into a development set and an evaluation set, which I will use to gauge the performance of all systems developed in the project.

Although the pronouns in the BBN corpus and Siddharthan's text have been co-indexed with their respective antecedents, the identities of these antecedents will not be used in development or evaluation: the purpose of the system is only to determine whether such an antecedent exists.

The three corpora are therefore of varying domains and it is uncertain how successfully any system will perform on one after being trained on another.

Chapter 2

Uses of the word 'it'

2.1 Tagging

In order to be able to tag different types of 'it' as referential (**Y**) and non-referential (**N**), it is necessary to develop a rigid definition of the contexts in which these different types occur. I have used the definitions of the various subtypes of non-referential 'it' given in this chapter to tag the texts of Siddharthan [2] in detail (see Ch. 1.4).

However, the BBN corpus is much larger and it would have been impractical for me to assess each occurrence manually. I used the antecedent coreference annotations of the BBN team to automatically tag each example with a binary **Y** or **N** decision, since these annotations do not contain information about the different subtypes of **Y** and **N** (although it would be possible to distinguish those **Y**s where the antecedent precedes the 'it' to those where it comes after it).

The Reuters corpus was not tagged at all, and is used in the experiments only as plain text.

I based the following classification of referential and non-referential 'it's loosely on those of Huddleston and Pullum [5] and Quirk *et al.* [6], but adapted it for ease of implementation into a computerised learning system.

In the classification below, I have cited examples of occurrences where they have been taken from a corpus. The remaining examples are of my own devising and represent elementary and commonly occurring uses of 'it' in English.

2.2 Y: Referential 'it'

Genuinely anaphoric 'it's fall roughly into two subclasses: **(x)** phrase referential and **(y)** eventuality or situation referential. In annotating Siddharthan's texts, I used the definitions of these classes given in this section as guidelines.

However, when tagging referential 'it's, I did not allocate **(x)** and **(y)** individual tags, but rather grouped the two together under **Y**. This was because the boundary between **(x)** and **(y)** forms a continuum, and it is difficult even for a human annotator to reliably distinguish these two classes. On the other hand, separating them both from non-referential usages of 'it' (the subtypes of **N**) is relatively straightforward.

(x) Phrase referential The antecedent of the 'it' is a well-defined noun or verb phrase.

> *But* ***the issue now***$_1$*, if I'm right, is not the simple one of American superpowerdom and whether Blair should be tailgating in its wake.* ***It***$_1$ *is whether, by internationalising the Iraqi conflict, Bush and Blair, with Chirac and Putin in acquiescence, will make the world a safer place.* [2]

This category includes **anticipatory anaphora**, a stylistic technique common in journalism.

> ***It***$_1$ *starts with a bottle of vodka or a wad of roubles stuffed into a lowly official's hand, but* ***corruption***$_1$ *goes right to the top in Russia...* [15]

(y) Eventuality or situation referential The antecedent of the 'it' cannot be easily pinpointed to a single phrase or, in some cases, a single sentence. The 'it' refers to the *situation* or general ideas established by a previous section of the discourse, and the sentence would still sound reasonably acceptable if the 'it' were replaced with 'this'.

> [*Garcia danced down the tee, waving wildly above his head as if he were about to attempt a world record hammer throw*]$_1$*. But he stopped and grinned:* ***it***$_1$ *was mock anger, as if he were taking the mickey out of himself, and he made strenuous, if unavailing, attempts to recover the situation.* [2]

The next example is even more convoluted, and presents considerable problems to an annotator—at what point should such cases be classified as non-referential of type **(iv)** (see Ch. 2.3)?

> [*Indians love the cinema and the Indian film industry, centred on Bombay, is one of the largest and most glamorous in the world. The vast proportion of films produced are gaudy melodramas based on three vital ingredients: romance, violence and music.*]$_1$ *You'll know what to expect from the fantastically hand-painted cinema billboards that dominate many streets. Imagine Rambo crossed with The Sound of Music and a Cecil B De Mille biblical epic, and you're halfway there.* ***It***$_1$*'s cheap operatic escapism, extremely harsh on the ears, and should not be missed.* [2]

In annotating Siddharthan's texts I have tried to be consistent and considered any instance which appears to link semantically to another section of text to belong to class **(y)**, although admittedly the above example falls on the borderline and really requires some real world knowledge from the annotator to establish the identity of the referent.

It should be pointed out, however, that the 'it' in the above example is still very different from the genuinely pleonastic usages of 'it' described in the next section: it has no place-holding function and is undoubtedly referential, yet identifying the referent poses some problems.

The question of real world knowledge, as is often the case in computational linguistics, brings us back to the **AI-complete problem** [7]. While this remains an issue, we cannot expect 100% accuracy even from a basic system for identifying pleonastic pronouns.

2.3 N: Non-referential 'it'

I have divided the types of 'it' which an anaphora resolution system should not attempt to resolve into five classes. More subdivisions are possible, but for the purposes of machine learning it is considered most practical to acknowledge these particular classes, since they have better defined boundaries than some other categorisations in the literature [5, 6].

(i) Extrapositional and impersonal 'it' This type of 'it', also known as *prop 'it'*, occurs only with a small class of verbs. The resultant sentences cannot be rearranged in an order that does not require an 'it'.

> ***It*** *seemed that things would never get any better.*
>
> $\star$ *That things would never get any better seemed.* [5]

This category intersects somewhat with category **(ii)** below. If an instance falls into both classes, the presence of an impersonal verb such as 'seem' takes precedence and the instance is tagged as **(i)**.

(ii) Nominal and sentential 'it'-cleft Both of these can be rearranged to a non-clefted form, although the rearrangement is sometimes not idiomatic. Sentential 'it'-clefts are often introduced by 'that'.
Nominal 'it'-cleft:

> ***It*** *was your father that was driving.*
> Rearranged: *Your father was driving.* [5]

Sentential 'it'-cleft:

> *Although* ***it*** *is possible (that) the story is being circulated by Western intelligence agencies to undermine morale within the regime, it seems likely there is much truth in these accounts.*
> Rearranged: *(⋆ Although) That the story is being circulated by Western intelligence agencies to undermine morale within the regime is possible, it seems likely...* [2]

Note that sentences such as

> ***It*** *really is quite astonishing that the committee that met this morning is finally turning its attention to the question of safety.* [2]

are classified as clefts **(ii)** rather than impersonal 'it' **(i)**, since a rearrangement is possible:

> *That the committee that met this morning is finally turning its attention to the question of safety really is quite astonishing.*

(iii) Infinitival 'it'-cleft, or extraposition The infinitival 'it'-cleft can take a subject introduced by 'for'. If the subject is not given, it may be possible to rearrange the cleft into a form that does not require 'it', although the result may sound somewhat unnatural.

> *It is hard (for the ordinary citizen) to know who can be trusted.*
> *(⋆ For the ordinary citizen) To know who can be trusted is hard...*

The relation of the non-clefted sentence to the equivalent infinitival 'it'-cleft is as follows [1]:

$$\text{subject} + \text{predicate} \approx \textit{it} + \text{predicate} + \text{subject}$$

For example,

> *To hear him say that + surprised me.*
> $\approx$ ***It*** *+ surprised me + to hear him say that.*

The 'it' does not need to be the subject of the main verb:

> *I find* ***it*** *impossible to believe that the mother knew nothing about what was going on in that house.* [18]

Infinitival 'it'-clefts are different from **'-ing'-cataphora**, which occurs mostly in spoken English. '-Ing'-cataphora cannot take a 'for'+subj complement and is marked as a 'do-resolve' case:

> *It's a dream come true, ($\star$ for him) winning this year.*

(iv) Weather, place, time, condition Here 'it' refers roughly to the weather or general situation (which was not explicitly mentioned). It is there as a syntactic placeholder.

> ***It*** *is raining.*
> ***It*** *is Monday.*
>
> *What a really frightful thing* ***it*** *would have been, if Mr. Jeremy had not noticed.* [2]
>
> ***It*** *is a crime when families are starving because...* [5]
> ***It*** *is a crime if families are starving because...*

Note that examples such as ***It*** *is a crime* ***that*** *families are starving...* are counted as sentential 'it'-cleft. This distinction reflects both the different syntactic structure and the subtle difference in meaning between the two sentences.

(v) 'It' in idioms Idiomatic 'it' occurs in relatively fixed phrases, with no enveloping structure.

Tagged corpus	**Y**	**N**				
	(x) or **(y)**	**(i)**	**(ii)**	**(iii)**	**(iv)**	**(v)**
Siddharthan Dev.	241	6	24	15	45	11
Siddharthan Eval.	238	34	16	15	31	18
BBN	1,935			9,831		

Table 2.1: The distribution of the various subclasses of 'it' in the tagged corpora used in the project.

*How's **it** going?*

*They made **it** to the quarterfinals.* [2]

*When **it** comes to free publicity...* [2]

2.4 Distribution of categories

After I tagged the occurrences of 'it' in Siddharthan's texts according to the above scheme, I calculated the frequency count for each category. These figures are shown in Table 2.1 for both Siddharthan's texts and with the BBN corpus for comparison. The figures for the BBN corpus were calculated from the number of 'it's that were co-indexed with any antecedent—since the BBN text was not tagged explicitly for subtypes of **N**, only the total figure for all non-resolvable 'it's is given.

2.5 Checking for consistency between corpora

In order to ascertain the feasibility of using Siddharthan's texts and the BBN corpora in the same experiments, when they have been tagged by different annotators and possibly under different tagging criteria, it is necessary first to check if the tagging schemes are consistent.

I randomly selected 100 examples from the BBN corpus, and used the tagging scheme given in this chapter to tag each one as either **Y**, or one of the non-referential cases **(i)** to **(v)**.

I then compared these with the tagging information that came with the BBN corpus, where each example is tagged as **Y** or **N** depending on whether a co-indexed antecedent exists in the BBN annotation.

Table 2.2 shows the confusion matrix for these two annotations. The agreement is 98%,

			Tags from BBN	
			Y	**N**
			(x) or **(y)**	**(i)** to **(v)**
My tags	**Y**	**(x)** or **(y)**	88 (5 clefts)	2
		(i)	0	0
		(ii)	0	2
	N	**(iii)**	0	3
		(iv)	0	2
		(v)	0	3

Table 2.2: A consistency check: the confusion matrix for 100 examples from the BBN corpus, showing my tags vs. the BBN team's tags. Note that while my tagset contains information about the subtypes **(i)** to **(v)** of non-resolve 'it's, this information is absent from the BBN tags.

meaning that there was disagreement between my tags and the BBN team's tags in only 2 out of 100 cases.

The BBN team found no suitable antecedent for these 2 anomalous cases, tagging them both as **N**, but my tagging scheme identified them as **Y**. The two cases are:

> *...after it emerges from bankruptcy proceedings* ***its rates will be among the highest in the nation***$_1$*, he said . "That attracts attention...* ***it***$_1$ *was just another one of the risk factors" that led to the company's decision to withdraw from the bidding.*

and

> *As individual investors have turned away from the stock market over the years, securities firms have scrambled* ***to find new products that brokers find easy to sell***$_1$*.*
>
> *And the firms are stretching their nets far and wide to do* ***it***$_1$*.*

These are both 'borderline' referential cases of subtype **(y)** (eventuality or situation referential), which was described in Ch. 2.2.

It is not surprising that the disagreements have appeared particularly with eventuality and situation referential 'it's, since these are the cases where a real-world understanding (and possibly some knowledge of finance) is often necessary to identify the antecedent. Referential cases of this type are characteristically difficult to classify.

Chapter 3

Parsing

3.1 Limitations of shallow processing

The majority of computational work in the literature on pleonastics and anaphora resolution in general has involved the use of relatively shallow features, such as the preceding and following words, template matches and other aspects of surface syntax, as input to the computer learning modules [1, 3, 4, 13].

This approach has its advantages, primarily its simplicity and the reduced need for understanding of the problem and for anticipation on the part of the programmer of the different types of examples that the system is likely to encounter.

However, the technique also has its disadvantages. An adverb or adjective inserted into a sentence does not change its underlying structure, but alters the surface syntax in such a way that many systems would easily fail to recognise a case which may be structurally very similar to examples in the training text. Part of my aversion to these shallow techniques is also that they are rather removed from what we know of natural language processing in the brain [19, 20].

I have therefore decided to use the RASP parser to obtain sentence structure trees and grammatical relations for all texts in this experiment.

3.2 The RASP system

RASP [21] is a parsing system for English that has been designed particularly for domain independence and robustness. It does not significantly underperform on texts that contain out-of-vocabulary words, making it an ideal choice for the varied domain texts dealt with in this project.

The system pipes text through stages of tokenisation, part-of-speech tagging, lemmatisation and parsing. In this experiment the tokenisation stage is sometimes bypassed as some corpora, such as BBN, have already been tokenised and split into sentences.

I have set the part-of-speech tagger to return the set of more probable tags for each word. These probabilities are propagated down to the parsing stage. The tagset used is a simplified version of the CLAWS C2 tagset [22].

The tagger output is then lemmatised and the lemmata and parts-of-speech are passed to the probabilistic parser, which analyses the chart of probable tags and generates a forest of possible parse trees with associated probabilities. From this it also constructs the set of n-best grammatical relations.

3.3 Parse charts and misparses

In order to investigate the reliability of RASP for the purposes of this project, I passed examples of all the different types of referential and non-referential 'it's mentioned in Ch. 2 through the parser and examined the 5-best output.

In the majority of cases, the top tree produced by RASP was the correct analysis. In a few cases, the system was thrown either by an adjective or adverb being wrongly interpreted and disrupting the rest of the tree, or by an unfamiliar word being tagged as the wrong part of speech. However, in these cases the correct analysis usually came with the second highest probability.

There were also a few occasions when the system failed to produce a complete tree, but this was rare.

I therefore decided to consider only the first analysis of each sentence when interpreting RASP's output. This way I could avoid having to propagate probabilities further through the

feature extraction stage, and greatly simplify the experiment.

Chapter 4

Feature extraction

4.1 The features

I wrote a Perl program to analyse the 1-best parse trees and grammatical relations produced by RASP and generate a grammatical feature vector which describes each occurrence of the word 'it' in the text.

The features that the program produces are listed in Appendix B. The features can be real numbers, Booleans, or strings.

The names of the string-typed features take the form **pattern-to-match** → ***property***; the Boolean-typed features have names of the form **pattern-to-match?**. I have allowed up to a maximum of 200 different string values for each string-typed feature. If more than 200 values occur in the data, the lower frequency values are grouped together as a single value, '*'.

4.2 Recursive analysis of RASP output

The program takes the RASP output of tokenised text, parse trees and grammatical relations, and works mainly from the grammatical relations to calculate the feature values. It scans through the tokens until it finds either an 'it' or an 'its' (the word 'it's' is actually two tokens, 'it' and ''s'). It then searches for any grammatical relations that involve this 'it', analyses all the lemmata these grammatical relations link to, and puts as much of the relevant information as possible

into a hash of features.

For example, the relationship of the 'it' to any verb in the sentence, such as **dobj** (direct object), is recorded in the feature **verb** → ***rel***.

The program then recursively examines any grammatical relations which link from these lemmata to any other lemmata in the sentence. Any lemma which is connected by a chain of up to 5 grammatical relations is analysed and if a corresponding feature exists, then the lemma, its POS and any inflections are stored in the features hash.

For example, if the 'it' is any kind of subject of the verb, and the verb also has an indirect object, then the lemma of this indirect object will be put in the feature **subj verb** → ***iobj***. The part-of-speech of the indirect object is put in the feature **subj verb** → ***iobj pos***, and so on.

This recursive examination of lemmata and relations is also used to determine whether the sentence containing the 'it' is structurally similar to any of the examples in Ch. 2.3. For example, the sentence

> *A ballooning federal debt, they say, will have made* ***it*** *impossible to deal with the needs of an aging population.* [2]

matches the pattern 'It is **ADJ** to **VP**', which corresponds to a particular type of infinitival 'it'-cleft **(iii)**. In this case the Boolean feature **Vs it ADJ to VP?** would be set to **TRUE** and the string-typed feature **Vs it ADJ to VP** → ***adj*** would be set to 'impossible'.

4.3 Word lists

The program also uses some pre-prepared word lists, which are given in Appendix C, in the pattern matching process. So if the word 'impossible' in the above example were replaced by 'difficult', which appears in the list of status adjectives in the appendix, then the Boolean feature **Vs it STATADJ to VP?** would also be triggered and set to **TRUE**.

4.4 Shallow syntactic features

The program also generates some other features which are not based on pattern matching or grammatical relations. These contain information about the preceding and following words and

distances to the previous and following noun phrases and sentence and paragraph boundaries. I will conduct some experiments to compare the effectiveness of these features to that of the grammatical structure based features.

4.5 Output

The program produces a table which contains the feature vector of each instance of 'it' in the document. The data is written to a file which can be passed to the Weka machine learning program (see next chapter) [23], which processes the values and builds or applies a data classifier.

If the text in question has already been annotated for pleonastics, an additional feature, **RESOLVE**, is included in each feature vector in the dataset. This takes values 'Y' or 'N', indicating the correct class which each example should be assigned to, and will be used by Weka to train the classifier.

Chapter 5

Data classifiers

5.1 Weka

The machine learning in this project is undertaken using the Weka data mining software [23]. The software provides a collection of algorithms for building data classifiers and for classifying unseen data.

I will investigate the performance of several different classifier algorithms that are available in Weka, namely decision trees and lists, an additive logistic regression algorithm, and a k-nearest neighbour classifier.

5.2 J48 decision tree

The first classifier I will investigate is J48, a Weka implementation of Quinlan's C4.5 decision tree algorithm [24]. The general approach of the C4.5 algorithm involves constructing and then optimising a decision tree [25, 26]:

1. Choose the feature which best differentiates the output feature values.

2. Create a separate tree branch for each value of the chosen feature. So if the feature is of type string, create a branch for every possible word value it can take. If it is numeric, differentiate the branches according to an equation or inequality condition. If it is Boolean,

create a branch for **TRUE** and one for **FALSE**.

3. Divide all the instances of 'it' into subgroups depending on their value for this feature.

4. Label any subgroup as terminal if either

   - all members in the subgroup have the same value for the chosen feature, or
   - there is only one node left in the subgroup, or no further distinguishing feature values can be determined.

   Label the current branch with a class (either **Y** or **N**) which applies to the majority of instances.

5. For each non-terminal subgroup, repeat the above process from step 1.

6. Simplify each rule in the decision tree by greedily deleting conditions in order to minimise the error rate.

7. Determine a default class (i.e. if no rule is applied), and delete any rules that will minimise the error rate.

C4.5 produces effective classifiers, but the process is complex and its runtime is $O(n^3)$ for n instances [27]. Despite the optimisation process (steps 5-7), rules are still restricted to conjunctions of feature value tests which occur along the decision tree path.

5.3 PART decision list

The PART algorithm [26] is inspired partly by the C4.5 approach. However, rather than deducing an initial set of rules or branches and discarding them in a separate optimisation stage, the PART algorithm produces the set of rules one at a time and does not require a global optimisation procedure.

PART uses a similar basic strategy, whereby a rule is built, the instances it covers are removed, and build new rules are then built from the reduced data set. The difference is the way that each rule is created: a pruned decision tree is constructed for the current set of instances, the leaf with the largest coverage is made into a rule, and the tree is discarded.

```
it-is-NP-S_np = UNDEF AND
it-is-ADJ-S_adj = UNDEF AND
subj-verb_xcomp = 'time': Y (43.0/1.0)
```

Figure 5.1: A PART decision rule: if the word 'time' is occurs with a certain RASP relation (**xcomp**) to the verb which the 'it' is the subject of, and some other relations are empty, then the 'it' is classified as referential (**Y**). This rule correctly classifies 43 of 44 training instances, giving a confidence weighting of $43/44 = 97\%$.

It may seem wasteful to produce and discard a whole decision tree just to obtain a single rule, but the process is optimised so that each tree is 'partial' (it contains branches to undefined subtrees). This enables the exploration of paths and rules that would not be considered by C4.5.

Each decision rule output by PART is also associated with a confidence weighting, defined as

$$\text{Confidence weight} = \frac{\text{Number of training instances correctly classified by the rule}}{\text{Number of training instances to which the rule applies}}. \quad (5.1)$$

For example, Fig. 5.1 shows a well-performing decision rule which is a conjunction of three conditions.

The PART method generally has runtime $O(an \log n)$ for n instances. In the worst possible case, where the number of rules increases linearly with n, the runtime is bounded by $O(an^2 \log n)$.

LogitBoost and PART have the advantage of being easily implemented into existing programs—this is useful as I am likely to need to process the output with my Perl program. It is also simple for a human to read and understand the classification rules.

5.4 Additive logistic regression

The LogitBoost algorithm [28] uses a technique called additive logistic regression, which is a way of combining several 'weak' classifiers to form a more powerful 'committee'.

For the two-class case, the algorithm involves weighting the instances in the training data according to how well the current set of classifiers performs on them.

1. Train each classifier on the training data, giving higher weight to instances that are currently misclassified.

2. Compute the expected performance over the training data.

3. Recalculate and renormalise the weights for all the training instances.

Additive logistic regression has the advantage that it almost never overfits to training data.

5.5 Nearest neighbours classifier

The other classifier I will investigate is IBk, a k-nearest neighbours classifier [29]. This involves comparing the feature vectors of instances that are to be classified to feature vectors already seen in the training data. I will vary the algorithm, so that instances are classified the same as the single nearest training vector, or the k nearest training vectors can be considered and weighted equally, or as $1 - \text{distance}$, or as $1/\text{distance}$.

Nearest neighbour techniques are simpler than the methods described earlier in that they do not involve the creation of abstract conditions, but rather compare instances directly to the training data. They perform very well on test data that is very similar to the training data, or when the training corpus is very large, or when the various cases to be classified are separate and unrelated entities. For this reason I suspect it may perform well on idiomatic non-referential usages of 'it' (type **(v)** described in Ch. 2.3), which would be difficult to abstract to a sequence of decision rules.

However, the method has the disadvantage that it generalises very badly and often misclassifies instances that do not closely resemble anything in the training data. It is also computationally intensive: the classification of one example requires consideration of every instance in the training data.

Chapter 6

Basic procedure

The training text, which has been annotated (either manually or automatically) for referential and non-referential 'it's, is split into sentences and parsed using RASP. The Perl program analyses the RASP output and extracts the feature vectors for each instance, including the additional feature **RESOLVE** which indicates whether each 'it' is referential or non-referential.

The resultant table of features for all the 'it's is used to train a Weka classifier. The instances in the test text are processed and converted into feature vectors by the Perl program, and the trained classifier is then used either directly through Weka, or via another Perl program, to classify these instances.

The performance of the system is then evaluated in terms of the proportion of instances in the test data that were classified correctly, as well as the confusion matrices and more complex analyses.

If the performance of the decision rules is considered adequate, they can be used to tag some instances automatically for further training (see Chs. 10-11).

Chapter 7

Baseline and upper bound accuracies

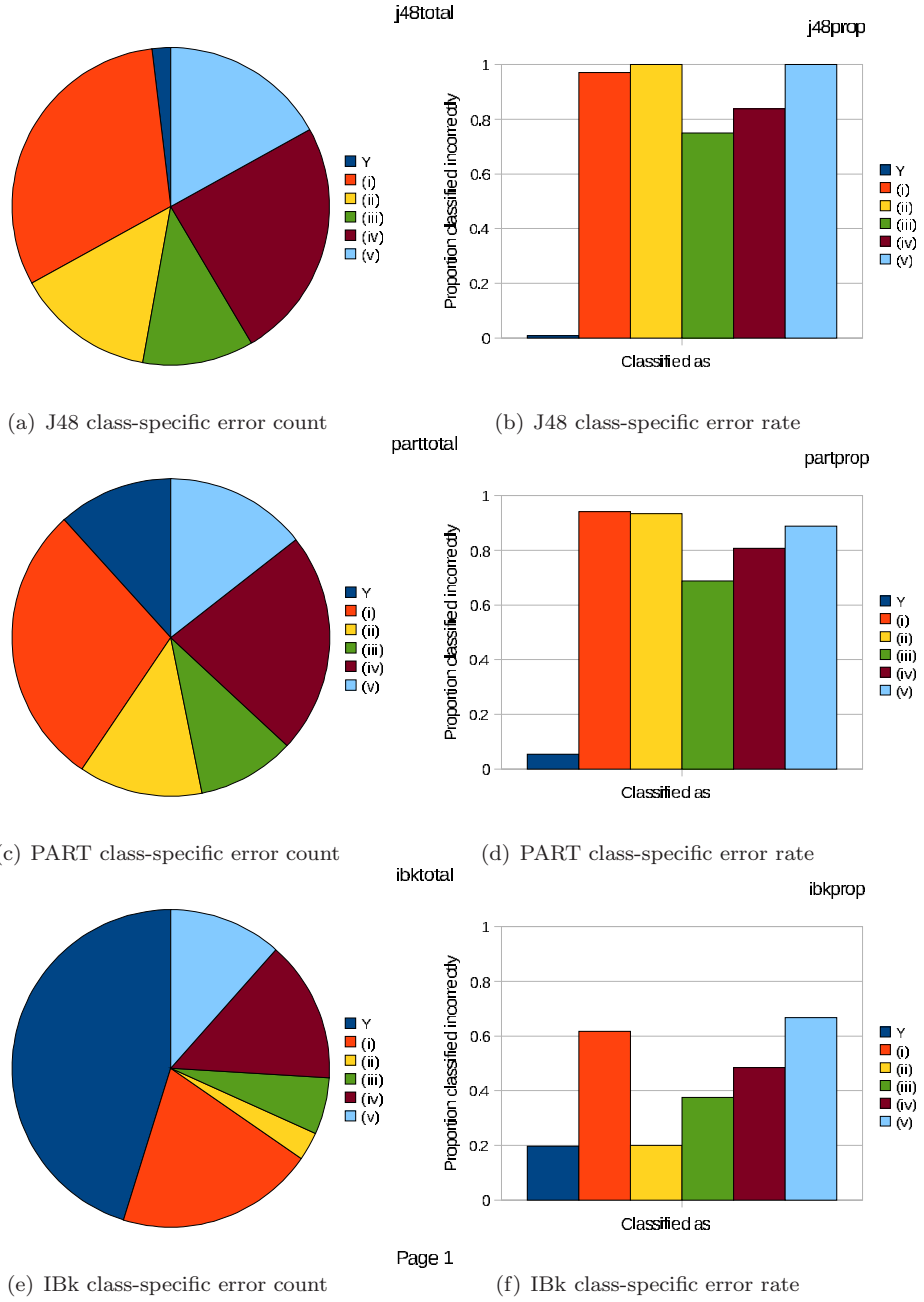
7.1 Baseline accuracy

The hand-tagged evaluation text from Siddharthan's work (see Table 1.1) contained 237 referential 'it's and 115 non-referential 'it's. The simplest solution to the resolve/don't resolve problem was to blindly classify all 'it's as referential. This gave an absolute baseline accuracy of $237/352 = \mathbf{67.3\%}$.

7.2 Upper bound accuracy

In order to calculate an upper bound for the performance of systems developed using unsupervised or semi-supervised techniques, the classifiers introduced in Ch. 5 (J48, PART, LogitBoost and IBk) were trained on Siddharthan's development text and tested on both the development and evaluation texts.

The performance of these various algorithms on Siddharthan's evaluation text are shown in Table 7.1. Figure 7.1 shows the class-specific performance of J48, PART and IBk on examples of type **Y** and **(i)** to **(v)**.



(a) J48 class-specific error count

(b) J48 class-specific error rate

(c) PART class-specific error count

(d) PART class-specific error rate

(e) IBk class-specific error count

(f) IBk class-specific error rate

Figure 7.1: The performance of three of the classifiers on class **Y** and subclasses **(i)** to **(v)** of **N** in Siddharthan's evalation text, in terms of the error count (pie charts) and the error rate as a proportion of instances misclassified (bar charts).

Page 1

Classifier	Description	Dev	Eval	SCV
J48	C4 decision tree [30]	75.4%	69.6%	74.6%
LogitBoost	Additive logistic regression [28]	78.4%	**71.6%**	72.2%
PART (0.25 conf. factor)	Decision list, separate-&-conquer [26]	75.4%	69.6%	**74.3%**
PART (0.5 conf. factor)		76.3%	70.5%	73.4%
IBk	k-nearest neighbours classifier [29]	**95.9%**	69.3%	69.3%

Table 7.1: The performance of each of the machine learning algorithms, trained on Siddharthan's development text, on both the development and evaluation texts. The stratified cross-validation (SCV) performance on the development set is also shown.

It is clear that the highest performance that we can expect to obtain from any type of classifier trained in any way and evaluated on some of these texts is around **96%**. This was achieved using the nearest neighbour classifier IBk trained and evaluated on Siddharthan's development text.

7.3 Evaluation of individual classifiers

The classifiers that performed better when trained on one half of the text and tested on the other half, and also when evaluated using stratified cross-validation, were the decision-based and regression-based classifiers J48, LogitBoost and PART, which all gave an accuracy of around 70% on the evaluation text. This is likely to be because they usually generalise better than nearest neighbour methods, and so are better equipped to cope with unseen situations using their knowledge of analogous instances in the development data.

Unfortunately the performance of these classifiers on the test data was only a few percent above the baseline performance of 67.3% (see Ch. 7.1). This suggests that much more training data is needed to produce a classification scheme even using any classifier.

Since J48 and PART are both based on the C4.5 method of constructing decision trees, it is interesting to note that they performed almost identically both on the evaluation text and in terms of stratified cross-validation. Fig. 7.1 shows that their relative performance on different types of instances is also comparable. Both J48 and PART classified nearly all of the **Y** cases correctly, and nearly all of the **N** cases incorrectly. The best performance on any type of non-referential 'it' was in both cases on type **(c)**, at around 30%. This shows that the decision tree algorithms tend to over-classify almost everything as **Y**, which is why the performance was only slightly above baseline (around 70% on Siddharthan's evaluation text and 74% cross-validation).

IBk performed better than the other classifiers when tested on its own training data, but worse than the others when tested on the evaluation text or by cross-validation. Figures 7.1(e) and 7.1(f) show that IBk outperformed the decision tree classifiers on all the non-referential categories of 'it's, but did worse on the **Y** category (a 20% error rate). This shows that IBk is probably doing a better job of classification than the other algorithms, but because there are many more **Y**s than **N**s in the test data, the 20% error rate on the **Y**s results in a lower accuracy for the whole test set.

The better performance of IBk on the **N**s may reflect the fact that the non-referential 'it's are probably more scattered in feature space, and it would be harder to find a concise pattern between them, whereas the referential 'it's are more homogeneous and a better match for a rule-based system.

The degradation in performance of IBk on unseen data is illustrated further in Figure 7.2, which shows how nearest neighbour methods can be very effective when all the instances in the test data strongly resemble those in the training data, but break down when the test instances become less familiar.

Because it generally performed slightly better than the other classifiers on held-out evaluation data and in stratified cross-validation, and also also because of the simplicity of implementing decision lists into the existing Perl code, I have decided to use the classifier PART for further development of the system. The aim of the project being to maximise the classification performance given only a limited amount of pre-tagged training text, I hope to obtain a significantly better accuracy using PART and a training corpus of comparable size to Siddharthan's development text.

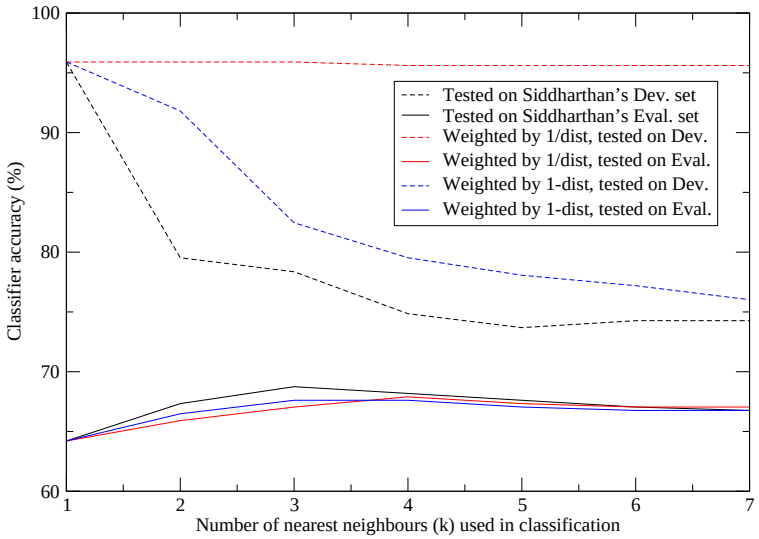


Figure 7.2: Performance of the IBk nearest neighbour classifier on Siddharthan's development and evaluation texts for three different weighting schemes (proportional, 1/distance and $1-$distance) and with a varying number of neighbours included in the classification. This illustrates the breakdown of k-nearest neighbour algorithms when confronted with unfamiliar data.

Chapter 8

Training on the BBN text

8.1 The classifier

I trained a PART classifier on the hand-tagged BBN corpus and tested it on Siddharthan's evaluation text.

First I built and evaluated the classifier using only syntactic and grammatical features, such as whether an 'it' matches the pattern 'it is **ADJ** that **S**' (see Appendix B). Then I added the features which related directly to the position of the 'it' in the sentence and its surrounding words, such as its distance from the start of the paragraph, and the rebuilt and re-evaluated the classifier.

8.2 The confidence weight threshold

For the evaluation process I introduced a **confidence weight threshold**. Decision rules are applied only when their confidence weight (see Ch. 5.3) exceeded a certain value between 0 and 1, the confidence weight threshold. The introduction of this threshold reduces the number of instances which the classifier attempts to classify (the **coverage** of the classifier), and may increase the accuracy on this smaller set of instances, but cannot increase the accuracy over the original set of instances.

Classifier feature set	CWT 0% actual performance	CWT 0% confusion matrix	CWT 100% actual performance	CWT 100% acc. on reduced coverage
Position independent (i.e. grammatical features)	69.3%	$\begin{pmatrix} 68 & 133 \\ 46 & 105 \end{pmatrix}$	65.3%	76.4%
Both position dependent and independent	70.7%	$\begin{pmatrix} 31 & 20 \\ 83 & 218 \end{pmatrix}$	53.1%	78.6%

Key to confusion matrices

$$\begin{pmatrix} \text{No. } \mathbf{N} \text{ classified as } \mathbf{N} & \text{No. } \mathbf{Y} \text{ classified as } \mathbf{N} \\ \text{No. } \mathbf{N} \text{ classified as } \mathbf{Y} & \text{No. } \mathbf{Y} \text{ classified as } \mathbf{Y} \end{pmatrix}$$

Table 8.1: Performance (accuracies and confusion matrices) of the PART classifier trained on the BBN corpus and tested on Siddharthan's evaluation text, with varying confidence weight thresholds (CWT), using a position independent (i.e. completely grammar-based) feature set and a position dependent feature set.

8.3 Implementation and results

The two classifiers were evaluated for no threshold and for a 100% confidence weight threshold. The results are shown in Table 8.1.

Although the position dependent features may seem somewhat crude, their inclusion in the classifier feature set significantly improved the performance of the classifier. It is interesting to note that when the 100% confidence weight threshold was introduced, the accuracy of the position dependent classifier was *reduced* significantly, even taking the reduced instance coverage into account.

In general, if the confidence weighting of a rule is correlated with its actual performance on the test set, one would expect the confidence weight threshold to improve the accuracy on the reduced set of instances. This is in fact what has happened in the case of both classifiers in Table 8.1, where the relative accuracies were increased to 76.4% and 78.6% by the reduction in coverage.

The unexpected reduction in accuracy is likely to be due to the difference between the writing styles in the BBN corpus and the cross-genre test data—it appears that information about a word's position in a sentence is a powerful feature for the automatically trained classifier, but degrades very quickly when ported to new genres.

I will investigate the difference in performance between these fundamentally different approaches over the next few chapters.

Chapter 9

Document-initial 'it's heuristic

9.1 Proposition

Since the majority of referential 'it's are not cataphoric, they require an antecedent to have appeared earlier in the text. This means that a non-cataphoric referential 'it' *cannot occur as the first word in a document.*

I propose that the binary feature of whether an 'it' is in document-initial position may be a useful heuristic for the development of a set of rules for identifying non-referential 'it's. Training based on this feature would have the advantage that it does not require any manually tagged data at all, and is therefore suitable for use on the Reuters corpus.

9.2 Sanity check

I manually examined a sample of 3047 documents in the Reuters corpus. The majority, 38, of the 50 of these documents which began with 'it' fell into category **N** according to the scheme in Ch. 2. These included

> ***It*** *still seems to be up in the air whether New Zealand First will announce this week or next week if it will go with Labour or National.*—class **(i)**: impersonal.

and

> ***It**'s probably easier to pick a mutual fund than it is to find the right financial advisor, but that doesn't stop most Americans from seeking professional advice.*—class **(iii)**: infinitival 'it'-cleft.

The remaining 12 document-initial 'it's turned out to be referential. These were all cases of anticipatory anaphora (see Ch. 2.2), mostly 'journalistic' uses such as:

> ***It***$_1$ *is now sipped as well as slammed and served in cognac snifters as well as shot glasses, and it can sell for up to $400 a bottle.* ***Tequila***$_1$*, once known as the mostly-to-be-avoided drink of party animals and macho revolutionaries, is brushing up its image.*—class **(x)**: phrase referential.

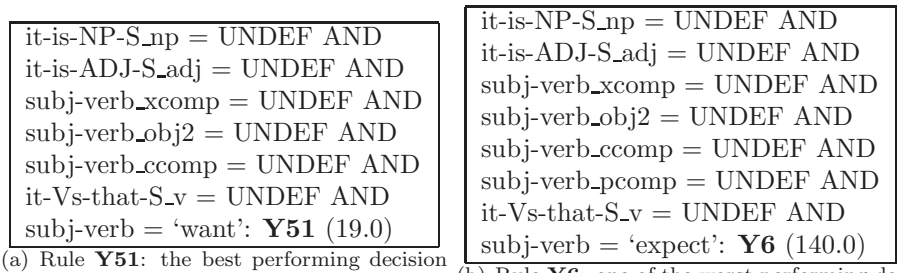
This shows that the position of the 'it' in the document is strongly correlated with whether or not it is referential. The question is whether this correlation is strong enough to be of use in unsupervised training. I will investigate the feasibility of this method and compare its performance with more traditional techniques based on manually annotated data.

9.3 Implementation and results

I automatically tagged the initial and final paragraphs in the Reuters corpus according to the initial 'it's heuristic. I tagged the instances of 'it' which occurred as the first word in a document as non-referential, and those occurring in the final 3 sentences of any document as referential.

I used the resulting 13,142 automatically tagged instances to produce a PART classifier consisting of 328 decision rules. This classifier gave an accuracy of 96.7% when applied to the training text.

However, when I evaluated the classifier on Siddharthan's multi-genre test data, it classified only 239 of 352 instances correctly, giving an accuracy of **67.9%**. Unfortunately this result is only slightly above the 67.3% baseline calculated in Ch. 7.1, and does not look particularly promising for the potential of the initial 'it's heuristic.



(a) Rule **Y51**: the best performing decision rule (correct: 4, incorrect: 1).

(b) Rule **Y6**: one of the worst performing decision rules (correct: 0, incorrect: 1).

Figure 9.1: A good decision rule and a bad decision rule (both giving a **Y** decision), assessed in terms of their performance on Siddharthan's training data. Both rules have 100% confidence weighting from the training data, although this is not a predictor of good performance on the evaluation data.

9.4 Identifying the causes of poor performance

It appeared that the bulk of the poor performance was due to a few particular rules in the classifier. In order to investigate where exactly the decision list was underperforming, I ran the classifie on Siddharthan's training set and examined the individual rules giving rise to each correct and incorrect decision. Two of the rules, **Y51** and **Y6**, are shown in Figure 9.1.

Rule **Y51** states that an 'it' should be marked as referential if it is the subject of the verb 'want'. This makes sense, since 'want' cannot occur in the English language with a pleonastic 'it' as its subject. The four instances which this correctly classified occur in the following sentence.

> ***It*** *doesn't want a smaller Israel,* ***it*** *wants no Israel;* ***it*** *doesn't want a reformed Saudi monarchy,* ***it*** *wants no Saudi monarchy.*

Similarly, rule **Y6** marks 'it's occurring as the subject of the verb 'expect' as referential. However, this rule misclassified the only instance in Siddharthan's training data for which it was invoked:

> *This new blood would,* ***it*** *was expected, make universities more efficient.*

The problem is that the verb 'expect' here is being used *passively*, and is actually a good example of a pleonastic co-occurring with a cognitive verb (see Appendix C). The instances in the Reuters corpus which gave rise to this rule were typically of the form

> ***It*** *had initally* [sic] *expected to price shares at between $9 and $11 each.*

where an 'it' appearing as the true (non-passive) subject of 'expect' is naturally referential. Since these instances occurred typically towards the end of a document, they were correctly tagged by the initial 'it's heuristic.

The problem here is one of *over-generalisation*. Verbs appearing in passive constructions which have not been seen by the classifier are automatically grouped with the corresponding active construction, which has been seen many times and was always tagged as referential. I observed this effect in the same experiment with other verbs, such as 'say', and will investigate this further.

9.5 Correcting for over-generalisation of passives

The observations above suggest that it may be productive to introduce a passive inversion procedure, so that an 'it' appearing as the grammatical subject of a passive verb would be treated as the object for the purposes of feature vector generation, and *vice versa*. This procedure essentially shifts the emphasis of the feature generation from a grammatical view on the sentence towards a more semantics-based view, and the change can be viewed as analogous to the development from the shallow features used in most of the literature (see Ch. 1.3) to the use of grammatical relations.

I altered my program in this way, so that, for example, values that were previously put in the attribute **subj verb → *iobj*** are now assigned to **non-subj verb → *iobj***, and those that were put in **non-subj verb → *iobj*** now go to **subj verb → *iobj***.

After this change, the system gave an accuracy of **67.3%** on Siddharthan's test text. This is not significantly different from the earlier performance in Ch. 9.3.

Unfortunately the introduction of the passive swapping procedure did not seem to significantly improve the performance. It was an interesting idea, since it considered the semantics of the text around the 'it', rather than just the syntactic constructions. However, it appears that the sparsity in the training data was a more significant hindrance to the performance than the issue of passive alternations.

This conclusion reflects the fact that pleonastic 'it's are a syntactic phenomenon which, although correlated with the semantics of a sentence, is fundamentally present to fill a syntactic

gap (i.e. to ensure that English verbs always have an explicitly stated subject).

9.6 Performance with confidence weighting

It is hoped that some of the underperforming rules can be identified at the training stage by their low confidence weights.

A variable confidence weight threshold (see Ch. 8.2) was introduced to see if it is possible to achieve a better performance on a limited section of the test set.

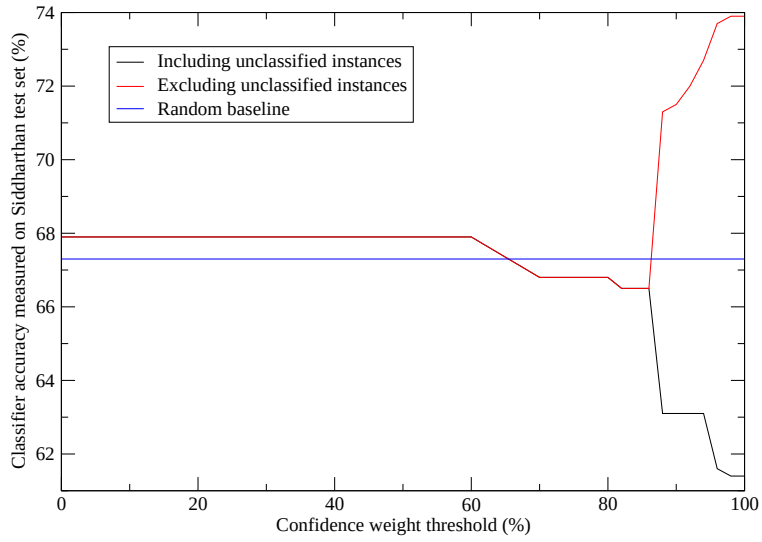
Fig. 9.2 shows the effect of varying the confidence weight threshold on the performance of the classifier on Siddharthan's test data. A confidence level of 100% reduces the coverage to 291, 82% of the original test set, but increases the accuracy to 215/291 = **73.9%**, almost 5% above baseline.

This performance, although only slightly worse than that of the position independent BBN-trained classifier (76.4%—see Ch. 8.3), is not promising. It appears that, as well as overgeneralising phenomena such as passive verb forms, the classifier has been trained to identify document-initial 'it's rather than non-referential 'it's. These two categories do not entirely overlap, and a large proportion of non-referential forms cannot occur in document initial position, such as any instances where the 'it' is the object, rather than the subject, of an active verb.

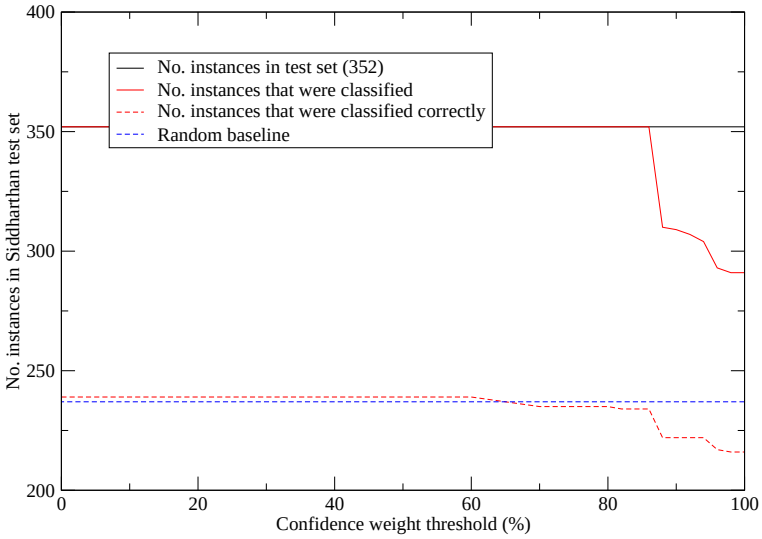
9.7 Predicting initial 'it's with BBN-trained classifier

In order to assess the compatibility of the BBN text annotations and the automatic annotation scheme of the initial 'it's heuristic, I evaluated the BBN-trained classifer from Ch. 8.3 on the Reuters text tagged according to the heuristic. This gave an accuracy of **58.4%**. The confusion matrix for this experiment is shown in Table 9.1.

It can be seen from the matrix that, while the initial 'it' heuristic has marked 96% of the instances as **Y**, the BBN classifier has classified roughly half of them as **Y** and half as **N**. There is also not much correlation between the BBN rules and the heuristic—a 58% agreement is not statistically significant.



(a) Classifier accuracy: instances correctly classified / instances examined (black line) or classified (red line).



(b) Classifier coverage.

Figure 9.2: Performance of the initial-'it'-developed PART classifier on Siddharthan's test data, with a varying confidence weight threshold applied.

No. of instances	Initial (i.e. **N**)	Non-initial (i.e. **Y**)	Rule accuracy
Classified as **N**	364	7,801	4.5%
Classified as **Y**	257	10,958	97.7%
Total performance			**58.4%**

Table 9.1: The confusion matrix for the classifer trained on the BBN text (Ch. 8.3) and tested on the initial-'it's-tagged data.

9.8 Discussion

The initial 'it's trained classifier failed to significantly outperform the baseline system, despite my efforts to optimise its performance and feature sets. The experiment in Ch. 9.7 also indicated that the rules generated from the BBN annotations do not reliably predict initial 'it's in the Reuters text. This strongly suggests that the initial 'it's heuristic, although an interesting idea, is not useful for augmenting tagged data. It is likely that, rather than training a classifier to find non-referential 'it's, my work in this chapter has merely produced classifiers that can find document initial 'it's—this is a somewhat circular result.

These findings suggest that the best way to proceed with developing a system using unsupervised methods is not to use a predetermined assumption such as the initial 'it's heuristic, but rather to use a small amount of hand-tagged data as a seed for a classifier, and augment this using techniques such as bootstrapping or co-training. I will investigate these approaches in the next chapters.

Chapter 10

Bootstrapping

10.1 Theory

Bootstrapping is an important technique in computer science whereby a simple system can be used to build a more complex system that serves the same purpose.

In computational linguistics problems, bootstrapping can be used to train a classifier on a tagged corpus, and then augment that corpus with automatically tagged instances which can be used to train a more advanced system [32].

Yarowsky [31] used bootstrapping to build an unsupervised word sense disambiguation algorithm which rivals the performance of techniques that require large amounts of hand-annotated data. This method exploited two properties of human language:

- One sense per collocation: nearby words give consistent clues to the sense of a target word.
- One sense per discourse: the sense of a target word is highly consistent within a document.

Yarowsky achieved an accuracy on unseen evaluation texts in excess of 96%, showing how two well-chosen assumptions can vastly reduce the need for human annotation.

Although the Yarowsky algorithm is extremely effective, it is relatively poorly understood mathematically. Abney [33] has analysed some variants of the algorithm and conjectures that it works to optimise an objective function, which is closely related to the likelihood of an instance.

10.2 The question of how to implement it

Unfortunately it is not immediately obvious what the equivalent crucial assumptions would be for a bootstrapping algorithm which identifies pleonastic pronouns.

I attempted to construct a bootstrapping system which would simply begin with a PART classifier trained on the BBN text, and would classify instances in the Reuters text according to this classifer. Examples from Reuters that are classified at a high confidence level are added to the BBN training data. The augmented training data can then be used to train a new classifier. The process is repeated iteratively.

Unfortunately, not only did this approach not improve the performance, but it was also based on flawed assumptions. Any rule that classifies some Reuters examples incorrectly quickly propagated through the iterations, acquiring a higher and higher confidence value.

I was not able to think of an equivalent for the pleonastics problem of Yarowsky's assumption that a word has the same sense throughout a document, that would enable Yarowsky's algorithm to be adapted for this project.

I therefore decided to investigate the potential of co-training, a variant of the basic boostrapping algorithm. This idea is introduced in the next chapter.

Chapter 11

Co-training

11.1 Introduction

Co-training is a variant of the basic bootstrapping procedure described in the last chapter. It is a partly supervised way of increasing the amount of annotated data available by making use of a small amount of manually labelled data and a large amount of unlabelled data [34].

The algorithm requires two classifiers, H_1 and H_2, based on *disjoint feature subsets* (mutually redundant perspectives on the input examples).

H_1 and H_2 are trained on a tagged data set, and used to tag some previously untagged data [35]. The most confident examples classified by each classifier are added to the training set of the other one. The training should progress (the performance should not get worse), since the new examples are at least as informative as random data.

11.2 The co-training algorithm

I based the co-training algorithm that I will implement in this project on the method of He and Gildea [35].

Constants

- k, the number of instances to add in each iteration.

Inputs

- An initial collection (**seed**) L of labelled instances. In this experiment I use Siddharthan's training text or the BBN corpus.
- An initial collection U of unlabelled instances. I use the Reuters corpus.
- Feature sets f_1 (word position related features) and f_2 (grammatical features) for classifiers H_1 and H_2.
- Training set $L_1^{(0)}$ for classifier H_1, $L_1^{(0)} = L$.
- Training set $L_2^{(0)}$ for classifier H_2, $L_2^{(0)} = L$.

Loop for $i = 1$ to I

- Train $H_1^{(i)}$ on $L_1^{(i-1)}$ by only considering only feature set f_1.
- Train $H_2^{(i)}$ on $L_2^{(i-1)}$ by only considering only feature set f_2.
- Use $H_1^{(i)}$ to annotate instances in U. Add the k instances with the highest confidence weight to $L_2^{(i)}$.
- Use $H_2^{(i)}$ to annotate instances in U. Add the k instances with the highest confidence weight to $L_1^{(i)}$.

Output at iteration i Train new classifier $H^{(i)}$ with feature set $f_1 \cup f_2$ on the training data $L_1^{(i)} \cap L_2^{(i)}$. The classifiers $H^{(i)}$, $H_1^{(i)}$ and $H_2^{(i)}$ can be used to classify new examples.

11.3 Co-training from Siddharthan's text

Initially I used Siddharthan's training text as the seed L. Figures 11.1 and 11.2 show the performance of the three classifiers H, H_1 and H_2 over 7 iterations with $k = 100$ and 20 respectively.

Unfortunately, neither of these methods appear to have improved the performance at all, although the performance of H_2 in particular fluctuates considerably over the iterations. It is possible that this is due to data sparsity in the seed set L: if a particular type of example is not seen at all in L, no amount of co-training can compensate for this.

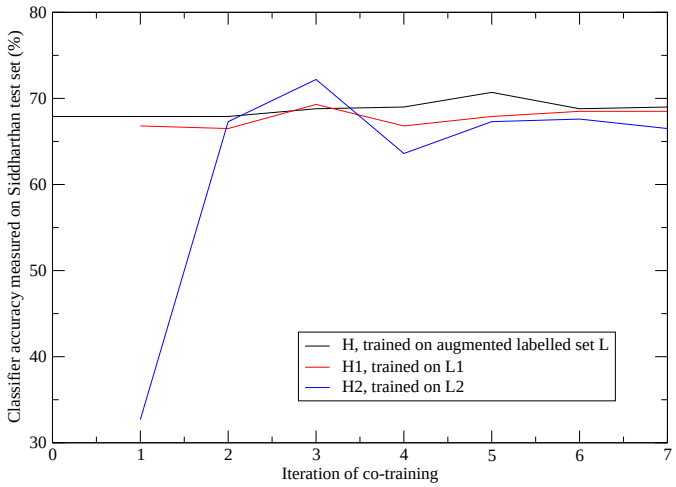


Figure 11.1: Co-training, starting from Siddharthan's data and adding $k = 100$ instances with each iteration.

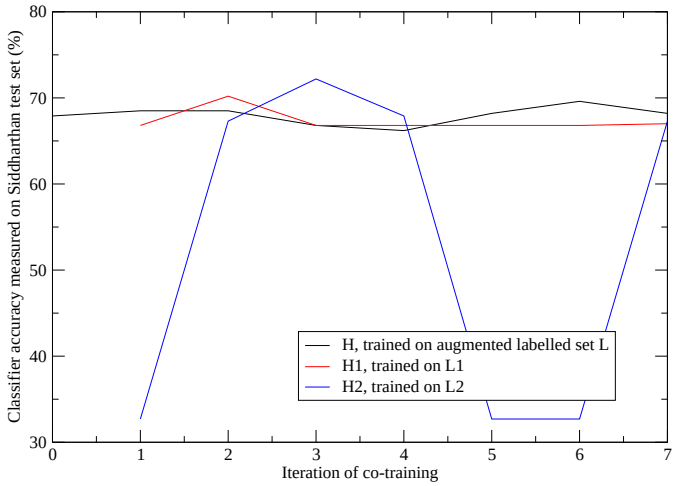


Figure 11.2: Co-training, starting from Siddharthan's data and adding $k = 20$ instances with each iteration.

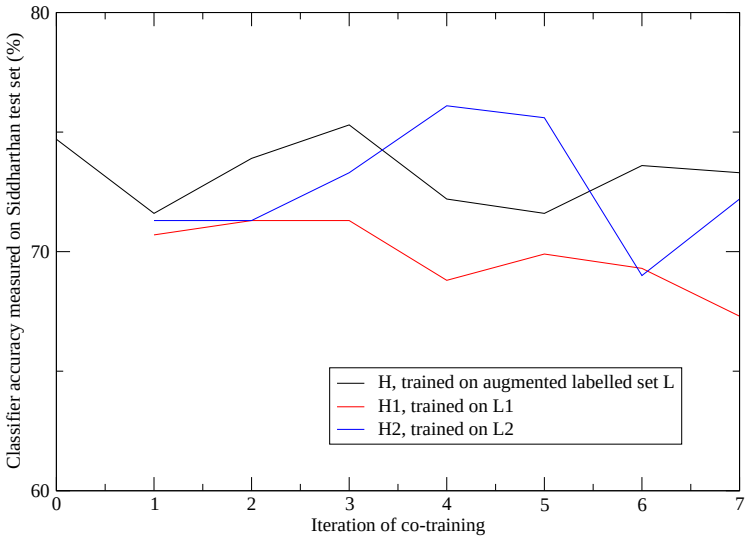


Figure 11.3: Co-training, starting from the BBN corpus and adding $k = 100$ instances with each iteration.

11.4 Co-training from the BBN corpus

Next, I used one half the BBN corpus as my seed set of examples L. Since this contained 5,883 examples, I was hoping to overcome the limitations of a small seed set like Siddharthan's text, and enable co-training to augment the performance. I used the same two classifiers H_1 and H_2, and executed 7 iterations of co-training. The results are shown in Figure 11.3.

Unfortunately, the co-training method does not appear to have improved the performance of the classifiers even with a very large initial set of annotated examples.

11.5 Discussion

The co-training methods attempted in this chapter failed to produce any improvement at all in classifier performance, although they induced some considerable fluctuations in accuracy over the course of the iterative process.

In my opinion, both co-training and basic Yarowsky-style bootstrapping are unlikely to be

capable of improving the performance of a RASP feature-based classifier for the pleonastics problem. This is probably due to a variety of factors, some of which were already mentioned in Ch. 10.

The environments in which non-referential 'it's can occur are so diverse that a classifier trained on the initial annotated text is unlikely to be reliable enough to augment the training set with any information that is both new and reliable.

Müller, Rapp and Strube [3] investigated the applicability of co-training for reference resolution, and also concluded that co-training cannot really reduce the amount of manual annotation and still produce a classifier with acceptable performance. They suggested that co-training may only be useful for a few very specialised purposes.

It appears that reference resolution and similar natural language problems are not well suited to this type of semi-supervised approach, at least with the current understanding of feature sets. A successful semi-supervised method for this problem would need either a more sophisticated algorithm than co-training, or a more advanced technique than my method of constructing feature vectors from RASP grammatical relations.

The construction of the PART classifier based even on these relatively small labelled sets was a very computationally intensive and time consuming process. This suggests that the primary limiting factor to the performance is the complexity of the classifiers and the availability of computer resources, rather than a shortage of manually annotated texts—although data sparsity is also an issue. I therefore do not consider bootstrapping techniques that increase the amount of training data to be a high priority for the pleonastics identification task.

Chapter 12

Final system

12.1 Combined PART and IBk classifier

The conclusions from the work already conducted are that semi-supervised methods are unsuitable for this task. The fundamental problems are the complexity of the phenomenon being investigated in comparison with the simplistic feature vector method, and also the sparsity of the data, even given the large quantities of annotated text available for training.

I have therefore decided to make use of all the annotated data available, and build a single classifier which combines the PART decision list algorithm with the IBk nearest neighbour methods to achieve optimal performance. With this method I hope to take advantage of my observations in Ch. 7.2 that PART performed well generally on **Y** cases, whereas IBk performed better on non-referential instances, especially classes **(ii)** and **(iii)**.

The training procedure for the new combined classifier is as follows:

1. Train PART classifier H_1 on Siddharthan's training text. Delete all rules with confidence level below a rule weight threshold of 90%.
2. Train PART classifier H_2 on the annotated BBN corpus.
3. Test H_2 on Siddharthan's training text and discard any rules which misclassify more than 10% of instances (but keep the rules which are not invoked, since Siddharthan's training text is small and most rules are not used to classify it).

4. Train IBk nearest neighbours classifier H_3 on all the instances in both Siddharthan's training text and the annotated BBN corpus, using the nearest 2 neighbours.

Using the combined classifier H_C to classify a set of unseen data is then a three-stage process:

1. Examine each instance in the data set,
   (a) Attempt to classify the instance using H_1.
   (b) If H_1 did not classify the instance, then attempt to classify it using H_2.
   (c) If H_1 and H_2 did not classify the instance, then classify it according to its nearest neighbours using H_3.

12.2 Results

I trained H_1, H_2 and H_3 on the relevant texts, and first tested them individually on Siddharthan's evaluation text.

The H_1 and H_2 PART classifiers classified only 147 of the 352 instances in the test data, but performed with an accuracy of 89.1%. Even given the reduced coverage, this is a definite improvement on the previous experiments with the confidence weight threshold.

The IBk classifier then classified the remaining 205 instances with an accuracy of 60.4%. If this was on the full set of 352 instances, an accuracy this low would be below baseline, but on the reduced coverage this reflects the fact that these 205 instances are characteristically difficult to classify, given that they were left unclassified by both H_1 and H_2.

The combined classifier H_C therefore gave an accuracy of **72.4%** on the entire test set of 352 instances. The accuracies and confusion matrices of these different stages are summarised in Figure 12.1.

The performance of H_1/H_2 and H_3 individually on the **Y** instances and different subtypes of **N** are graphed in Fig. 12.2. Fig. 12.3 shows the performance of the complete classifier on these subtypes.

I have included the rules used by the PART classifiers H_1 and H_2 in Appendix A.

		True class N	Y
Classified as	N	21	9
	Y	7	110

(a) H_1 & H_2: cov. 147/352, acc. 89.1%.

		True class N	Y
Classified as	N	58	26
	Y	55	66

(b) H_3: cov. 205/352, acc. 60.4%

		True class N	Y
Classified as	N	79	35
	Y	62	176

(c) H_C: cov. all 352, acc. 72.4%

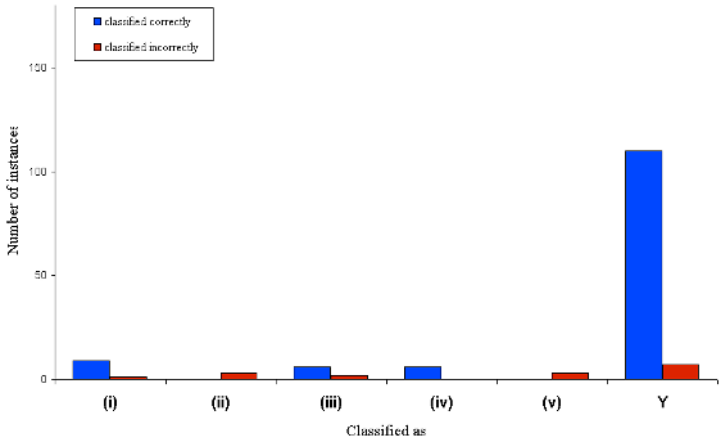
Figure 12.1: Confusion matrices, accuracies and coverage figures for the PART classifiers H_1 and H_2, the IBk classifier H_3, and the PART/IBk classifier H_C, which is a combination of H_1, H_2 and H_3.

12.3 Discussion

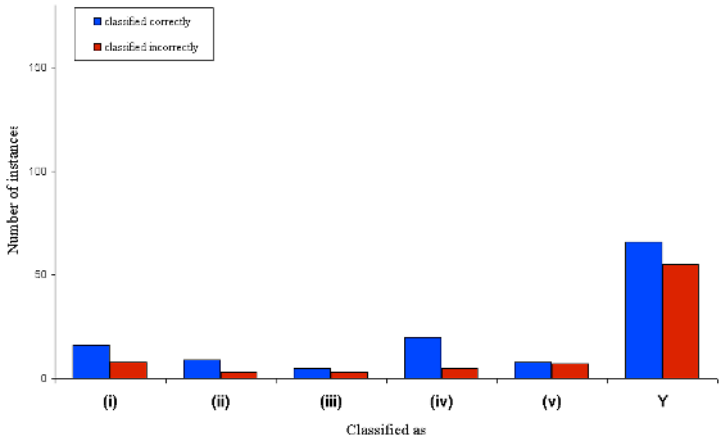
Although the accuracy of 72.4% is marginally above baseline, I do not consider this to be a positive result for completely supervised pleonastic identification methods. It appears that the examples that the PART classifiers found difficult to classify accurately are also the examples that IBk had trouble with.

It can be seen from the confusion matrices that the performance would have been improved if the IBk stage of the combined classifier were replaced by a simple baseline system that classifies all instances automatically as **Y**.

I had thought that, since IBk seemed to perform slightly better than PART on certain cases, notably **(ii)** and **(iii)**, it might be effective to partition the classifying task between these two methods, in order that each could contribute to the performance on the instances which it is good at. However, it appears that this does not work. IBk may perform better than PART on idiomatic cases, but for this to become useful I think it would be necessary to have even more training data and build a very large classifier—a computationally intensive task on the scale I am working on at the moment.



(a) H_1 and H_2 combined.



(b) H_3.

Figure 12.2: The accuracy of the individual classifier components of H_C on different types of instances in Siddharthan's test data.

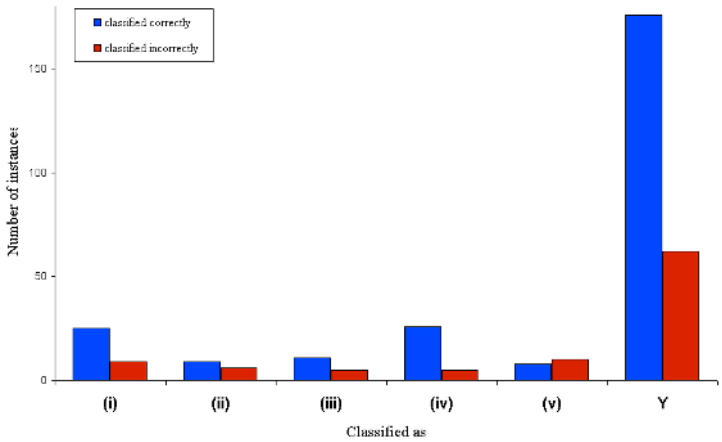


Figure 12.3: The accuracy of the combined classifier H_C on different types of instances in Siddharthan's test data.

Chapter 13

Conclusions

I have developed a basic system for identifying pleonastic uses of the word 'it' in English text, using machine learning algorithms which are trained on a corpus of manually annotated text and based on a feature set derived from a RASP parse of the text.

I attempted to develop some methods of training this system using semi-supervised methods with unannotated text. I investigated the possibility of working from 'it's which occur at the beginning of a document and are unlikely to have a genuine antecedent. I then investigated some bootstrapping approaches. The results of these innovations were negative: there was no improvement in the system performance.

I then attempted to improve my performance by using and combining different types of classifiers (notably decision lists and nearest neighbour methods). Unfortunately this did not bring any significant improvement in accuracy either.

Throughout the project, the best accuracies my system achieved on the cross-genre Siddharthan test set were all in the region of 75%. A baseline system which classifies all instances automatically as referential achieved an accuracy of 69%.

However, in Ch. 12.2 I manage to achieve an 89.1% accuracy on a limited portion ($147/352 = 41\%$) of the test set, using a PART classifier developed from Siddharthan's training data and the BBN annotations. This, and my other experiments on reduced coverage portions of the test set, illustrate how the classifiers tend to perform quite reliably on a small fraction of the problems, and also that the confidence weights for individual rules in a PART classifier are a reasonably

reliable indication of how well the rule can be expected to perform on the test data.

Although much of the work in the literature on pleonastic pronouns (Ch. 1.3) did achieve higher accuracies than this, I should point out that many of the authors were tackling a slightly different problem. My task is made considerably more difficult by the fact that while my test data spans a wide variety of genres, my training data is nearly all single-domain (news or business). All of the authors I researched in Ch. 1.3 were working on very limited domain text, and at least one group (Paice and Husk [1]) evaluated their system on the data that they had used for development, giving them a large advantage in terms of performance.

The performance of the baseline system and of some of the more successful rules in the PART classifiers showed that it is often the easiest option to assume that a pronoun is referential, since the referential cases seemed fill up most of the feature space, with the non-referential cases scattered throughout. This observation is backed up by the rules in my PART classifiers: in general they consisted of a long list of specific cases which are to be classified as **N**, with a default (if no other rule is matched) to classify the instance as **Y**.

This would indicate that a nearest neighbour algorithm is the ideal approach to the problem, but unfortunately this method did not perform any better than the decision lists.

It is perhaps fortunate that **Y** turned out to be the easiest case to classify. A practical application of this algorithm would probably involve its output being passed to an anaphora resolution procedure, with information whether or not to resolve a particular instance. In such cases it would be necessary to 'err on the side of caution', i.e. if the pleonastics classifier was sure about a particular case, it would be safest to classify it automatically as a referential pronoun. The anaphora resolution algorithm could decide later whether there really is an antecedent that is worth resolving.

It is interesting to note that my PART classifiers almost never invoked the word lists in Appendix C. This is perhaps a consequence of data sparsity, or possibly because the word lists are small and not comprehensive. It could also indicate that pleonastic usages are not as dependent on fixed classes of words as is generally indicated in the literature [5, 6].

I think that further research on this topic would require the use of very large corpora of cross-domain training data, with more computing resources. More theoretical research into different types of pleonastics would also be productive.

Chapter 14

Acknowledgements

Thanks to my supervisor, Ann Copestake, for being very accessible and helpful throughout this project, and for her detailed comments on the first draft of this report.

Appendix A

Final system PART classifier

These are the rules used by the PART classifiers H_1 and H_2 in the final system developed in the project.

it-Vs-ADJ-to-VP? = FALSE AND
subj-verb_xcomp = UNDEF AND
non-subj-verb_obj2 = UNDEF AND
subj-verb_ccomp = UNDEF AND
this-word = its: Y

it-Vs-ADJ-to-VP? = FALSE AND
subj-verb_xcomp = UNDEF AND
subj-verb_ccomp = UNDEF AND
non-subj-verb_obj2 = UNDEF AND
non-subj-verb_ncsubj = UNDEF AND
subj-verb_ncmod = UNDEF AND
subj-verb = UNDEF AND
verb_rel = dobj: Y

it-Vs-ADJ-to-VP? = TRUE: N

subj-verb_xcomp = UNDEF AND
non-subj-verb_ncmod = UNDEF AND
subj-verb_ccomp = UNDEF AND
non-subj-verb_ncsubj = UNDEF AND
subj-verb_ncmod = UNDEF AND
prev-word = UNDEF AND
dist-to-last-sing-NP-in-prev-sentence& <= 2: Y

subj-verb_xcomp = UNDEF AND
non-subj-verb_ncmod = UNDEF AND
subj-verb_ccomp = UNDEF AND
subj-verb_ncmod = UNDEF AND
non-subj-verb_ncsubj = UNDEF AND
it-Vs-ADJ-S? = TRUE: Y

subj-verb_xcomp = UNDEF AND
non-subj-verb_ncmod = UNDEF AND
subj-verb_ccomp = UNDEF AND
subj-verb_ncmod = UNDEF AND
non-subj-verb_ncsubj = UNDEF AND
prev-word = that: Y

subj-verb_xcomp = time: N

non-subj-verb_ncmod = UNDEF AND
subj-verb_xcomp = UNDEF AND
subj-verb_ccomp = UNDEF AND
subj-verb_ncmod = UNDEF AND
non-subj-verb_ncsubj = UNDEF AND

prev-word = but AND
dist-from-start-of-para& <= 110: N

non-subj-verb_ncmod = UNDEF AND
subj-verb_xcomp = UNDEF AND
subj-verb_ccomp = UNDEF AND
subj-verb_ncmod = UNDEF AND
non-subj-verb_ncsubj = UNDEF AND
subj-verb_iobj = to: Y

non-subj-verb_ncmod = UNDEF AND
subj-verb_xcomp = UNDEF AND
subj-verb_ccomp = UNDEF AND
subj-verb_ncmod = UNDEF AND
non-subj-verb_ncsubj = UNDEF AND
prev-word = UNDEF AND
dist-from-start-of-para& <= 131: N

subj-verb_xcomp = UNDEF AND
non-subj-verb_ncmod = UNDEF AND
subj-verb_ccomp = UNDEF AND
subj-verb_ncmod = UNDEF AND
non-subj-verb_ncsubj = UNDEF AND
prev-word = ',': N

subj-verb_xcomp = UNDEF AND
non-subj-verb_ncmod = UNDEF AND
subj-verb_ccomp = UNDEF AND
subj-verb_ncmod = UNDEF AND
non-subj-verb_ncsubj = UNDEF AND

prev-word = on: Y

subj-verb_xcomp = UNDEF AND
non-subj-verb_ncmod = UNDEF AND
subj-verb_ccomp = UNDEF AND
subj-verb_ncmod = UNDEF AND
non-subj-verb_ncsubj = UNDEF AND
prev-word = UNDEF: Y

subj-verb_xcomp = UNDEF AND
non-subj-verb_ncmod = UNDEF AND
subj-verb_ccomp = UNDEF AND
subj-verb_ncmod = UNDEF AND
subj-verb = UNDEF AND
non-subj-verb_ncsubj = UNDEF AND
prev-word = and AND
next-word = was: N

subj-verb_xcomp = UNDEF AND
non-subj-verb_ncmod = UNDEF AND
subj-verb_ccomp = UNDEF AND
subj-verb_ncmod = UNDEF AND
subj-verb = UNDEF AND
non-subj-verb_ncsubj = UNDEF AND
prev-word = and AND
dist-to-last-sing-NP-in-sentence& > 2: Y

this-word = its AND
word-number& ¿ 2 AND
next-word = * AND

prev-word = of: Y

this-word = its AND
word-number& ¿ 2 AND
next-word = * AND
prev-word = in AND
word-number& ¿ 3: Y

this-word = its AND
prev-word = during: Y

this-word = its AND
next-word = latest: Y

this-word = its AND
dist-to-last-sing-NP-in-sentence& <= 41 AND
next-word = new: Y

this-word = its AND
dist-to-last-sing-NP-in-sentence& <= 41 AND
next-word = sales: Y

this-word = its AND
dist-to-last-sing-NP-in-sentence& <= 41 AND
next-word = restructuring AND
dist-to-last-sing-NP-in-prev-sentence& <= 10: Y

this-word = its AND
prev-word = from: Y

this-word = its: Y

it-is-ADJ-S_adj = UNDEF AND

non-subj-verb_ncmod = UNDEF AND

non-subj-verb_xcomp = UNDEF AND

subj-verb_xcomp = UNDEF AND

subj-verb_ccomp = UNDEF AND

it-Vs-ADJ-S_adj = UNDEF AND

it-Vs-that-S_v = UNDEF AND

subj-verb_ncmod = UNDEF AND

subj-verb_dobj = UNDEF AND

subj-verb = UNDEF AND

non-subj-verb_ncsubj = UNDEF AND

prev-word = before AND

dist-to-last-sing-NP-in-prev-sentence& <= 13: Y

Appendix B

Grammatical features

The grammatical features extracted by the Perl program, along with some example values which these features can take, are presented in this Appendix. Here **NP** and **VP** refer to noun and verb phrases respectively, and **WV** etc. refer to word categories (such as weather verbs) which can be found in Appendix C.

Groups of features marked with a * are position-related features, which are treated as a subgroup of features for the purposes of the initial BBN experiment (Ch. 8) and as the feature set for classifier H_1 in the co-training experiments (Ch. 11).

B.1 Properties relating to the position in the sentence*

distance from start of paragraph : *real*

previous word : UNDEF, ,, and, that, but, ", of, to, ;, on, buy, although, think, when, get, make

next word : was, is, 's, ., would, C, to, does, in, has, did, attention, a, on, must, into, the, had

word number : *real*

B.2 Inter-sentence and inter-noun-phrase distances*

dist to last sing NP in sentence : *real*

dist to next sing NP in sentence : *real*

dist to last sing NP in prev sentence : *real*

B.3 Grammatical properties of the 'it'

this word : it, its

verb → *rel* : UNDEF, ncsubj, dobj, ncsubj-dobj, det, obj2, dobj-ncsub

subj verb : UNDEF, be, have, expect, say, take, make, plan, sell, go

subj verb → *pos* : UNDEF, VVD, VBZ, VV0, VVZ, VVN, VBDZ, VVG, VB0, VHZ, VBN

subj verb → *infl* : UNDEF, s, ed, X, ing, en, NULL

subj verb → *dobj* : UNDEF, and, $, share, agreement, loss, or, it, sale, plan

subj verb → *dobj pos* : UNDEF, NN1, NN2, CC, NNU, NP1, MC, PPH1, NN, PPHO2, DD1

subj verb → *dobj infl* : UNDEF, X, s, NULL, ing, ed

subj verb → *iobj* : UNDEF, to, in, for, with, of, on, by, from, as, at, about

subj verb → *iobj pos* : UNDEF, II, IF, IW, IO, CSA, CC, RGQ

subj verb → *iobj infl* : UNDEF, X

subj verb → *obj* : UNDEF, which, what, and, option, number, provision

subj verb → *obj pos* : UNDEF, NN1, DDQ, NN2, CC, NNU, NN, PNQO, DDQV, DAR, PNQS

subj verb → *obj infl* : UNDEF, X, s, NULL

subj verb → *obj2* : UNDEF, share, "%", $, -rrb-, and, or, day, right, epo

subj verb → *obj2 pos* : UNDEF, NN1, NN2, NP1, NNU, CC, RT, NNJ, MC, NNT1, NN, NNJ1

subj verb → *obj2 infl* : UNDEF, X, s

subj verb → *xcomp* : UNDEF, hard, clear, difficult, and, possible, time, one

subj verb → *xcomp pos* : UNDEF, JJ, NN1, VVG, II, NNT1, JJR, CC, NN2, VVN, NP1, MC1

subj verb → *xcomp infl* : UNDEF, X, ing, s, ed, NULL, en

subj verb → *ncmod* : UNDEF, not, also, in, year, of, for, just, now, on

subj verb → *ncmod pos* : UNDEF, RR, XX, II, NNT1, RT, IO, IF, RRR, RA, RG, ICS, IW

subj verb → *ncmod infl* : UNDEF, X, NULL, s

subj verb → *cmod* : UNDEF, if, when, while, although, because, as, though

subj verb → *cmod pos* : UNDEF, CS, ICS, II, CSA, CSW, IF, CC, CCB, IW

subj verb → *cmod infl* : UNDEF, X

subj verb → *xmod* : UNDEF, include, as, say, give, and, while, accord, by

subj verb → *xmod pos* : UNDEF, VVG, VVN, CSA, VVD, II, CC, ICS, CS, IF, IW

subj verb → *xmod infl* : UNDEF, ing, X, ed, en

subj verb → *pmod* : UNDEF, because, as, so, even, at, to, where, down

subj verb → *pmod pos* : UNDEF, CS, CSA, II

subj verb → *pmod infl* : UNDEF, X

subj verb → *aux* : UNDEF, be, will, have, would, do, could, may, can, might

subj verb → *aux pos* : UNDEF, VM, VBZ, VHZ, VBDZ, VB0, VDZ, VHD, VDD, VBN, VH0

subj verb → *aux infl* : UNDEF, X, s, ed, NULL, en, ing

subj verb → *pcomp* : UNDEF, in, for, like, because, as, worth, on, at, when

subj verb → *pcomp pos* : UNDEF, II, IF, CS, CSA, ICS, IO, CSN, RGQ, CSW, IW

subj verb → *pcomp infl* : UNDEF, X

subj verb → *ccomp* : UNDEF, be, when, if, in, say, and, expect, for, because

subj verb → *ccomp pos* : UNDEF, VV0, VVN, CS, VVZ, II, VVD, VBZ, VVG, CC, VBDZ, IF

subj verb → *ccomp infl* : UNDEF, X, ed, s, ing, NULL, en

subj verb → *arg* → *mod* : UNDEF, when, how, in, why, for, of, where, at, on, under

subj verb → *arg* → *mod pos* : UNDEF, RRQ, II, RGQ, IF, IO, CC, ICS

subj verb → *arg* → *mod infl* : UNDEF, X

subj verb → *passive?* : TRUE, FALSE

non subj verb : UNDEF, make, do, say, take, call, put, have, give, find

non subj verb → *pos* : UNDEF, VV0, VVD, VVG, VVZ, VVN, VD0, VH0, VBZ, VDG, VHD

non subj verb → *infl* : UNDEF, X, ed, ing, s, en, NULL

non subj verb → *ncsubj* : UNDEF, they, i, we, he, you, and, that, it, who, she

non subj verb → *ncsubj pos* : UNDEF, NN1, NP1, NN2, PPHS2, PPIS1, PPIS2, PPHS1, CC

non subj verb → *ncsubj infl* : UNDEF, X, s, NULL, ed

non subj verb → *dobj* : UNDEF, it, what, and, some, protection, penalty, price

non subj verb → *dobj pos* : UNDEF, PPH1, NN1, DDQ, CC, MD, DD, PPY, PPHO1, NN2

non subj verb → *dobj infl* : UNDEF, X, NULL, s

non subj verb → *iobj* : UNDEF, to, in, with, as, for, on, into, from, of, at

non subj verb → *iobj pos* : UNDEF, II, IW, CSA, IF, IO, RGQ

non subj verb → *iobj infl* : UNDEF, X

non subj verb → *obj* : UNDEF, what, month, way, which, time

non subj verb → *obj pos* : UNDEF, NNT1, DDQ, NN1

non subj verb → *obj infl* : UNDEF, X

non subj verb → *obj2* : UNDEF, way, -lcb-, $, or, all, crime, right, look

non subj verb → *obj2 pos* : UNDEF, NN1, NP1, NNU, CC, DB, DA1, PPHO1, VVG, NNU2

non subj verb → *obj2 infl* : UNDEF, X, s, NULL, ing

non subj verb → *xcomp* : UNDEF, difficult, easier, hard, possible, as, harder

non subj verb → *xcomp pos* : UNDEF, JJ, JJR, VVG, CSA, II, CC, DA1, IO, CS, DAR, MD

non subj verb → *xcomp infl* : UNDEF, X, ing

non subj verb → *ncmod* : UNDEF, not, in, just, also, of, and, with, to, even, year

non subj verb → *ncmod pos* : UNDEF, RR, XX, II, NNT1, CC, IO, RP, IW, RA, IF, RG, RRR

non subj verb → *ncmod infl* : UNDEF, X, NULL, s

non subj verb → *cmod* : UNDEF, if, when, while, as, because, once, until, before

non subj verb → *cmod pos* : UNDEF, CS, ICS, CSA, IF, CC, II

non subj verb → *cmod infl* : UNDEF, X

non subj verb → *xmod* : UNDEF, seek, accord, by, now, ask, include, while, allow

non subj verb → *xmod pos* : UNDEF, VVG, CS, IF, VVN, II

non subj verb → *xmod infl* : UNDEF, ing, X, ed

non subj verb → *pmod* : UNDEF, when, away, over, as, until, because, at

non subj verb → *pmod pos* : UNDEF, II, CS, ICS, CSA

non subj verb → *pmod infl* : UNDEF, X

non subj verb → *aux* : UNDEF, would, will, be, can, have, do, could, may, might

non subj verb → *aux pos* : UNDEF, VM, VH0, VD0, VBZ, VHZ, VBR, VDZ, VDD, VBDZ

non subj verb → *aux infl* : UNDEF, X, NULL, s, ed, en, ing

non subj verb → *pcomp* : UNDEF, to

non subj verb → *pcomp pos* : UNDEF, II

non subj verb → *pcomp infl* : UNDEF, X

non subj verb → *ccomp* : UNDEF, in, go, be, when, like, to, if, because, say, come

non subj verb → *ccomp pos* : UNDEF, CS, VV0, II, VVD, VVZ, VVN, VVG, VBZ, VDG, IF

non subj verb → *ccomp infl* : UNDEF, X, ed, s, ing

non subj verb → *arg* → *mod* : UNDEF, how

non subj verb → *arg* → *mod pos* : UNDEF, RGQ

non subj verb → *arg* → *mod infl* : UNDEF, X

non subj verb → *passive?* : TRUE, FALSE

B.4 Derived grammatical properties

'It's raining', etc.

it WVs? : TRUE, FALSE

'I don't like it when you behave like this'

it Vs when S? : TRUE, FALSE

Covering some example constructions in [8]

it is ADJ that S? : TRUE, FALSE

it is ADJ that S → *adj* : UNDEF, clear, true, possible, important, apparent

it is NP that S? : TRUE, FALSE

it is NP that S → *np* : UNDEF, accident, reminder, case, coincidence

it is MADJ that S? : TRUE, FALSE

it is STATADJ that S? : TRUE, FALSE

it is STKADJ that S? : TRUE, FALSE

it Vs ADJ that S? : TRUE, FALSE

it Vs ADJ that S → *adj* : UNDEF, clear, true, possible, unlikely, likely

it Vs ADJ that S → *v* : UNDEF, be, become, make, think, appear, find

it MVs ADJ that S? : TRUE, FALSE

it MVs MADJ that S? : TRUE, FALSE

it Vs NP that S? : TRUE, FALSE

it Vs NP that S → *np* : UNDEF, $, accident, year, platform, reminder

it Vs NP that S → *v* : UNDEF, be, give, make, accuse, produce, disclose, say

it MVs NP that S? : TRUE, FALSE

it is ADJ S? : TRUE, FALSE

it is ADJ S → *adj* : UNDEF, too, very, good, also, just, only, still, time, so

it is NP S? : TRUE, FALSE

it is NP S → *np* : UNDEF, time, exercise, activity, possibility, visit

it Vs ADJ S? : TRUE, FALSE

it Vs ADJ S → *adj* : UNDEF, new, too, good, very, only, also, just, so, even

it Vs NP S? : TRUE, FALSE

it Vs NP S → *np* : UNDEF, $, acquisition, agreement, evidence, time

it is ADJ to VP? : TRUE, FALSE

it is NP to VP? : TRUE, FALSE

it Vs ADJ to VP? : TRUE, FALSE

it Vs NP to VP? : TRUE, FALSE

it is ADJ for NP? : TRUE, FALSE

it is ADJ for NP → *adj* : UNDEF, crucial, appropriate, ridiculous, easy

it is ADJ for NP to VP? : TRUE, FALSE

it is ADJ for NP to VP → *adj* : UNDEF, crucial, appropriate, ridiculous, easy

it Vs ADJ for NP? : TRUE, FALSE

it Vs ADJ for NP → *adj* : UNDEF, crucial, appropriate, ridiculous, easy

it Vs ADJ for NP to VP? : TRUE, FALSE

it Vs ADJ for NP to VP → *adj* : UNDEF, crucial, appropriate, ridiculous, easy

it is NP for NP? : TRUE, FALSE

it is NP for NP → *np* : UNDEF

it is NP for NP to VP? : TRUE, FALSE

it Vs NP for NP? : TRUE, FALSE

it Vs NP for NP → *np* : UNDEF

it Vs NP for NP to VP? : TRUE, FALSE

it is Ved that S? : TRUE, FALSE

it is Ved that S → *v* : UNDEF, estimate, expect, know, understand, hope, argue

it is COGVed that S? : TRUE, FALSE

it Vs that S? : TRUE, FALSE

it Vs that S → *v* : UNDEF, be, expect, have, make, plan, say, agree, sell

it MVs that S? : TRUE, FALSE

Vs it ADJ to VP? : TRUE, FALSE

Vs it ADJ to VP → *adj* : UNDEF, difficult, hard, easier, possible, harder

Vs it STATADJ to VP? : TRUE, FALSE

Vs it STKADJ to VP? : TRUE, FALSE

Vs it NP to VP? : TRUE, FALSE

Vs it STATNP to VP? : TRUE, FALSE

Vs it ADJ for NP to VP? : TRUE, FALSE

Vs it STATADJ for NP to VP? : TRUE, FALSE

Vs it STKADJ for NP to VP? : TRUE, FALSE

B.5 Decision

RESOLVE : Y, N, C

Appendix C

Word groups

Word categories that are likely to occur often with expletive 'it's and are therefore prepared for the project.

C.1 Verbs of being

be look remain get stay become

C.2 Cognitive verbs

'it is **COGV**ed that **S**'. Based on Lappin and Leass [8] and Paice and Husk [1].

recommend think believe know anticipate assume
expect wonder doubt question understand

C.3 Modal adjectives

'It is **MODALADJ** that **S**', 'It is **MODALADJ** (for **NP**) to **VP**'. Based on Lappin and Leass [8].

necessary possible certain likely important good
useful advisable convenient sufficient economical easy
desirable difficult legal fortunate

C.4 Modal verbs

'It **MV**s that...'.

look seem appear happen

C.5 State-of-knowledge adjectives

From Paice and Husk [1].

debatable questionable uncertain clear doubtful dubious unclear unknown

C.6 Status adjectives

'It is **STATADJ** to/that...'. Paice and Husk [1].

abnormal	bad	common	dangerous	easier	faster
foolish	good	hard	impracticable	incumbent	interesting
necessary	obligatory	preferred	rare	relevant	safe
shock	sufficient	tempting	uncommon	unsafe	useful
wise	wrong	advantageous	beneficial	correct	easiest
fastest	advisable	best	customary	difficult	easy
feasible	harder	important	inadvisable	infeasible	irrelevant
justified	normal	hardest	impossibility	inappropriate	intended
practicable	rarer	remains	safer	simple	
traditional	unhelpful	unscientific	useless	wiser	rarest
right	safest	simpler	trivial	unnecessary	unusual
usual	wisest	amazing	idle	startling	astonishing
logical	surprising	disappointing	profitable	exciting	rational
possible					

C.7 Status nouns

'It is a **STATN** to/that...'. Based on Paice and Husk [1].

job policy truism problem

C.8 Weather verbs

'It **WV**s', 'It is **WV**ing'. From Levin [36].

blow	clear	drizzle	fog	freeze	gust
hail	howl	lightning	mist	mizzle	pelt
pour	precipitate	rain	roar	shower	sleet
snow	spit	spot	sprinkle	storm	swelter
teem	thaw	thunder			

Bibliography

[1] C. D. Paice and G. D. Husk, *Towards the automatic recognition of anaphoric features in English text: the impersonal pronoun "it"*. In *Computer Speech and Language* **2**, 109-32 (1987).

[2] A. Siddharthan, *Syntactic Simplification and Text Cohesion*. PhD Thesis, University of Cambridge (2003).

[3] C. Müller, S. Rapp, M. Strube, *Applying Co-Training to Reference Resolution*. In *Proceedings of the 40th Annual Meeting of ACL*, Philadelphia (2002).

[4] R. Evans, *Applying Machine Learning Toward an Automatic Classification of It. Literary and Linguistic Computing* **16**, 1 (2001).

[5] R. Huddleston and G. K. Pullum, *The Cambridge Grammar of the English Language*. Cambridge University Press (2002).

[6] R. Quirk, S. Greenbaum, G. Leech and J. Svartvik, *A Comprehensive Grammar of the English Language*. Longman (1985).

[7] J. R. Hobbs and R. C. Moore (eds.), *Formal Theories of the Commonsense World*. Ablex Pub, CA (1985).

[8] S. Lappin and H. J. Leass, *An Algorithm for Pronominal Anaphora Resolution. Association for Computational Linguistics* **20**, 4 (1994).

[9] M. C. McCord, *Slot Grammar: A System for Simpler Construction of Practical Natural Language Grammars*. In *Proc. Int. Symposium on Natural Language and Logic*, ed. R. Studer. Springer-Verlag, London (1990).

[10] Univ. Lancaster, Univ. Oslo, and Norwegian Computing Centre for the Humanities (Bergen), *Lancaster-Oslo/Bergen Corpus of British English*. Dept. of English, Univ. Oslo (1978).

[11] G. Sampson, *English for the Computer: the SUSANNE Corpus and Analytic Scheme*. Oxford University Press, Oxford (1995).

[12] L. Burnard, *Users Reference Guide British National Corpus Version 1.0*. Oxford University Computing Services, Oxford (1995).

[13] C. Müller, *Automatic Detection of Nonreferential It in Spoken Multi-Party Dialog*. In *European ACL* (2006).

[14] A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke and C. Wooters, *The ICSI Meeting Corpus*. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Hong Kong (2003).

[15] Reuters, *Reuters Corpus*, Volume **1**: *English language*, 1996-08-20 to 1997-08-19 (released 2000).

[16] Linguistic Data Consortium, *BBN Pronoun Coreference and Entity Type Corpus* (annotated Wall Street Journal text). Univ. of Pennsylvania (2005).

[17] Guardian Media Group, *The Guardian*, `www.guardian.co.uk/` (accessed 13/06/2008).

[18] Times Newspapers Ltd., *The Times*, `www.timesonline.co.uk/` (accessed 13/06/2008).

[19] C. Vogel, U. Hahn, H. Branigan, *Cross-Serial Dependencies Are Not Hard to Process*. In *Proc. 16th conference on Computational linguistics* **1**, Copenhagen, Denmark (1996).

[20] A. Radford, M. Atkinson, D. Britan, H. Clahsen and A. Spencer, *Linguistics: an introduction*. Cambridge University Press (1999).

[21] E. Briscoe, J. Carroll and R. Watson, *The Second Release of the RASP System*. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, Sydney, Australia (2006).

[22] University Centre for Computer Corpus Research on Language, Lancaster University, *UCREL CLAWS2 Tagset*, `ucrel.lancs.ac.uk/claws2tags.html` (accessed 16/06/2008).

[23] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd Ed., Morgan Kaufmann, San Francisco (2005).

[24] J. R. Quinlan, *Improved Use of Continuous Attributes in C4.5*. In *J. Artificial Intelligence Research* **4**, 77-90 (1996).

[25] Gamma Ray Burst Research at Minnesota State University, Mankato, *Manual*, `grb.mnsu.edu/grbts` (accessed 16/06/2008).

[26] E. Frank and I. H. Witten, *Generating Accurate Rule Sets Without Global Optimization*. In Shavlik, J. (ed.), *Machine Learning: Proceedings of the Fifteenth International Conference*, Morgan Kaufmann Publishers, San Francisco, CA (1998).

[27] W. W. Cohen, *Fast effective rule induction*. In *Proc. 12th Int. Conf. on Machine Learning*, Morgan Kaufmann (1995).

[28] J. Friedman, T. Hastie and R. Tibshirani, *Additive Logistic Regression: a Statistical View of Boosting*. Technical report, Stanford University (1998).

[29] D. Aha and D. Kibler *Instance-based learning algorithms. Machine Learning* **6**, 37-66 (1991).

[30] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers (1993).

[31] D. Yarowsky, *Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. Association for Computational Linguistics* **95**, 189-196 (1994).

[32] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall (2000).

[33] S. Abney, *Understanding the Yarowsky Algorithm*. Univ. of Michigan (2004).

[34] A. Blum and T. Mitchell, *Combining Labeled and Unlabeled Data with Co-Training*. In *Proc. 11th Annual Conference on Computational Learning Theory*, Madison, WI (1998).

[35] S. He, D. Gildea, *Self-training and Co-training for Semantic Role Labeling: Primary Report*. Univ. of Rochester Comp. Sci. Dept. Tech. Report 891, Rochester, NY 14627 (2004).

[36] B. Levin, *English Verb Classes and Alternations: A Preliminary Investigation*. Univ. Chicago Press, Chicago and London (1993).

[37] J. Alba-Salas, *Lexically Selected Expletives: Evidence from Basque and Romance*. *SKY J. Linguistics* **17** pp35-100 (2004).

[38] C. Callison-Burch, *Co-training for Statistical Machine Translation*. MSc thesis, Univ. of Edinburgh Division of Informatics (2002).

[39] J. Carroll, A. Copestake, R. Malouf and S. Oepen, *LKB*. `http://wiki.delph-in.net/moin/LkbTop` (1991-2004).

[40] A. Copestake, *Robust Minimal Recursion Semantics*. Computer Laboratory, University of Cambridge (2004).

[41] C. Fellbaum (ed.), *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. MIT Press, Cambridge, MA (1998).

[42] D. Flickinger, E. Bender, A. Copestake, R. Malouf and S. Oepen, *LinGO English Resource Grammar*. `http://www.delph-in.net/erg/` (accessed 2008).

[43] M. A. Hearst, *Noun homograph disambiguation*. In *Proceedings of the 7th Annual Conference of the University of Waterloo Centre for the New OED and Text Research*, Oxford (1991).