CONSTRAINTS ON AUXILIARY DELETION IN COLLOQUIAL WELSH

Florian Breit

School of Linguistics & English Language, Bangor University

Academic Year 2011/2012

Banner ID: 500231618

CONSTRAINTS ON AUXILIARY DELETION IN COLLOQUIAL WELSH

A dissertation submitted in partial
fulfilment of the requirements for
the degree of BA (Hons) in Linguistics

By
Florian Breit
School of Linguistics & English Language
Bangor University

PRIFYSGOL
BANGOR
UNIVERSITY

Submitted on
21st May 2012

Do not be too timid and
squeamish about your actions.
All life is an experiment.
The more experiments you
make the better.

Ralph Waldo Emerson

Contents

List of Tables

List of Figures

List of Program Code Listings

List of Abbreviations

1P	First person plural
1S	First person singular
2P	Second person plural
2S	Second person singular
3P	Third person plural
3S	Third person singular
3Sm/3Sf	Third person singular male/female
AAVE	African American Vernacular English
AD	Auxiliary Deletion
AFF	Affirmative mood
Aux	Auxiliary verb
CSV	Comma separated values (file format)
D/Det	Determiner
FUT	Future tense
HTML	Hypertext Markup Language (file format)
IMPFV	Imperfective aspect
INFL	Inflection
INT	Interrogative mood
NEG	Negation, Negative mood

NP	Noun Phrase
O	Object
PAST	Past tense
PD	Particle Deletion
PFV	Perfective aspect
PHP	PHP: Hypertext Preprocessor (scripting language)
PN	Proper Noun
POS	Positive
Prep	Preposition
PRES	Present tense
PRT	Particle
Q	Question particle
S	Subject
SQL	Structured Query Language
V	Verb
V1	Verb first

Acknowledgements

I would like to express my thankfulness to the staff at the School of Linguistics & English Language, who at large have offered me many an open ear and ample platform for discussion often far beyond the curriculum. No doubt the changes the department went through while I studied for this degree have been dramatic, and indeed traumatic at times, but they have also been a constant provider of opportunity to see things from different perspectives, which at this early stage in a hopeful linguist's career has been most useful. Special thanks here foremost go to my personal tutor, Peredur Davies, who has been outstanding and inspiring in every aspect and who is ultimately responsible for getting me hooked on the topic discussed in this dissertation (diolch am yr holl bysgod!), but also Dirk Bury, Marco Tamburelli and Paul Carter for their open doors, insightful discussion of theory and inspiration.

Specifically in creation of this work I would also like to thank my supervisor, Margaret Deuchar, who has provided many useful resources in the inital stages of research for my dissertation and to the dissertation co-ordinator, Vicky Chondrogianni, who has readily offered help with my dissertation when it was needed. Further thanks go to my fellow aspiring linguist Rhian Davies, who has been very patient with my attempts at learning her native language and always lent her expertise as a native informant, for this dissertation and other work prepared during the degree course. Diolch yn fawr!

I have to also extend my gratefulness to all my other Bangor friends

and acquaintances, my Welsh teachers at Lifelong Learning, and to Bangor Linguistics Society, who have all helped pass time and worries all along the way and made these last three years a great experience. Thank you!

Last but not least I have to thank my parents, without whose support – both emotionally and financially – all this would not have been possible and for whom my respect and admiration has grown in proportion to distance from home: Merci viilmols Mama un Papa!

Declaration

I hereby declare that this dissertation is my own work in partial fulfilment of requirements for *BA (Hons) Linguistics*.

Florian Breit

Bangor, Gwynedd

21st May 2012

1 Introduction

Some recent research on the grammatical properties of the spoken varieties of modern colloquial Welsh have described a phenomenon known as Auxiliary Deletion (AD), through which some clause-initial auxiliaries in these varieties can be omitted. This phenomenon has as of yet not received much individual attention apart from Jones (2004) and Davies (2010), where AD is the subject of a large part of their overall studies; Davies and Deuchar (in preparation) are further currently preparing a somewhat more elaborated version of the investigation into AD presented in Davies (2010).

Welsh being a verb-initial language, auxiliary constructions in Welsh are traditionally associated with periphrastic constructions of the type AuxSVO, which stand in contrast to so called synthetic constructions of the type VSO, with a finite initial verb. This type of construction is the one primarily employed in colloquial varieties of Welsh and stands in relation to colloquial Welsh exhibiting only limited final verb morphology for verbs other than auxiliaries, especially so with the present tense (for more on periphrastic and synthetic constructions see Borsley et al., 2007). Further, Davies (2010, p. 285) has shown that in his corpus analysis of AD with the second person singular pronoun *ti*, 92.75% of the sentences analysed featured AD, suggest-

ing that AD in these constructions is the norm in spoken colloquial Welsh and it can consequently be argued that this phenomenon clearly deserves some more in-depth treatment than it has so-far received.

While Jones (2004) limits himself to offering his observations and intuitions on what grammatical person AD may occur with, Davies (2010) focuses only on the second person singular, for which he has performed a corpus analysis as part of his PhD study. In the second chapter of this dissertation I will look at the previous literature on AD, focusing on the studies by Jones (2004), Davies (2010) and Davies and Deuchar (in preparation). Following this, in the second chapter I undertake to test Jones' (2004) and Borsley et al.'s (2007) proposed limitations on grammatical person and AD by carrying out a corpus study on the conversational Welsh *Siarad* corpus that has been collected at Bangor University over the last years (Deuchar et al., 2009; see also Davies, 2010, pp. 150–178 for a description of the methodology applied in creating the corpus). On the basis of this I will then present an experimental study testing the acceptability of AD in some of the grammatical configurations in which Welsh auxiliaries can be found (e.g. clause-initial, Wh-questions with movement and in-situ, with a preposed subject/object, in coordinated clauses, etc.) in the third chapter. From this it is hoped to gain some insight into the possible constraints that may apply to AD beyond that of grammatical person or number proposed by Jones (2004), which I will discuss in chapter four. This will be followed by my conclusion in the final chapter. It is believed that identification of such constraints can form the basis of further investigation offering greater insights on the processes underlying AD in Welsh. As such the aim of this

dissertations is exploration: to establish some basic information about the syntactic conditions under which AD may occur, in order to lay the ground work for some further theoretical work tackling this phenomenon.

2 Existing Literature on Auxiliary Deletion

2.1 Introduction

Auxiliary Deletion in Welsh has generally not been much discussed before. As mentioned already in the introduction, the two principal sources of description until now are from Jones (2004) and Davies (2010), though the phenomenon itself has previously and since been variously acknowledged (e.g. King, 1996; Borsley et al., 2007).

Davies' (2010) study primarily looks at AD as an indicator of contact-induced language-change. In doing so, he also makes reference to a previous explicit description of AD from Phillips (2007), written in Welsh, in which Phillips stipulates that AD may indicate a shift in word order from VSO/AuxSVO to SVO, an analysis which Davies (2010) agreed with, arguing that this is due to convergence towards English SVO word order stemming from the extensive bilingualism present in Wales[1]. They posit

[1]There are normally believed to be no monolingual Welsh speakers in Wales above the age of 3 years any more. This can be partially attributed to the nature of bilingualism and presence of English in the media, which exposes speakers to at least some English

that AD occurs due to language contact between Welsh and English and reflects language change in Welsh, where Welsh gradually tends to assume a surface structure similar to that of English SVO, thus leading to a preference in AuxSVO structures with deleted auxiliaries which resemble these. As the relations of AD to language contact and language change are not of much interest apart from the fact and extent to which AD appears to presently occur in Welsh, I will not discuss this matter much further in this chapter, though I will draw on Davies (2010) and Davies and Deuchar (in preparation) in section 2.3, where following an overview of the Welsh auxiliary system in general in section 2.2, I further discuss the extent to which AD is known to currently occur in Welsh and give some real examples of AD in Welsh.

Other than Davies (2010) and Phillips (2007), Jones (2004) and Borsley et al. (2007) do not present any analysis of real data containing AD but instead limit themselves to stating for which grammatical persons AD may occur. This will be further discussed in section 2.4 and later forms the basis for my corpus study in chapter 3, in which I test their intuitions on the correlation of AD and grammatical number.

Similar phenomena to AD in Welsh are also known to occur in other languages, for instance in African American Vernacular English (AAVE) and in Central Salish as already referred to by Davies (2010). However

anywhere in Wales, but is especially due to English being part of the National Curriculum for Wales (see DfES, 2008) (note however that Key Stage 1 in Welsh-medium schools is exempt from this requirement, see ACCAC, 2000, p. 2). However, there is no actual comprehensive data on this available, and while the UK Census in Wales features a question on Welsh language ability, it does not contain any questions to record an ability to speak English, so that there is no potential to draw data on monolingual Welsh speakers from this.

while noting that work on AD in these languages is also largely of a broad descriptive type, I will not discuss these phenomena in detail here due to space constraints.

Section 2.5 presents a summary of the literature reviewed in this chapter.

2.2 Auxiliaries and Auxiliary-initial Clauses in Colloquial Welsh

As a verb-initial language Welsh sentence structures are often divided into two basic constructions: the synthetic construction and the periphrastic construction. Synthetic constructions are of the type VSO with an initial finite verb. Periphrastic constructions are of the basic type AuxSVO with an initial auxiliary and an infinitive verb after the subject, although auxiliaries derived from *bod* 'to be' also require use of an aspectual particle which is placed between the subject and its following infinitive verb (Borsley et al., 2007, pp. 38–47). While literary Welsh makes extensive use of the synthetic construction in all tenses, in colloquial Welsh constructions of the periphrastic type are much more common. One reason for this is that in either variety of Welsh there is no specific inflectional paradigm for the present tense apart from that for *bod* 'to be'. Instead there is one inflectional paradigm which in formal Welsh represents both the present and the future tense, but which is only used for the future tense in colloquial Welsh. In colloquial Welsh the periphrastic construction with a present tense paradigm for the auxiliary *bod* (in conjunction with an aspectual particle such as the

Table 2.1: Paradigm of *bod* in a northern variety of colloquial Welsh.

	future	present	past
1S	*bydda*	*dw*	*ôn*
2S	*byddi*	*wyt*	*oeddet*
3S	*bydd*	*ydy/mae*	*oedd*
1P	*byddwn*	*dan*	*oeddwn*
2P	*byddwch*	*dach*	*oeddech*
3P	*byddwn*	*ydyn/maen*	*oedden*

imperfective *yn* or the perfective *wedi*) is used to mark the present tense instead (cf. Borsley et al., 2007, 9–12). Additionally to the present tense[2], in colloquial Welsh the periphrastic construction is very common for past tense statements, often with the auxiliary *gwneud* 'to do'.

Borsley et al. (2007, p. 38) themselves define auxiliaries as "certain verbal elements which appear with a verbal complement of some kind and allow the expression of a meaning which would be expressed by a single verb in some languages." As such their definition of auxiliaries itself relies heavily on the above distinction between synthetic and periphrastic constructions, and it is probable that as such any initial tensed verb in a periphrastic construction is an auxiliary (see also Borsley et al., 2007, pp. 44–47 for a further discussion of possible syntactic tests for auxiliary status). Following from this, Borsley et al. (2007, p. 38) describe three types of auxiliary-initial clauses: aspectual clauses (i.e. those headed by the auxiliary *bod* and containing an aspect marker), *gwneud*-clauses and *ddaru*-clauses.

[2]Note that with the use of aspect this covers both what equates in English to the present tense and the simple past. For instance both "I eat" and "I am eating" can be expressed as the imperfective present tense statement "Dw i'n bwyta" and "I ate" can

Table 2.2: Paradigm of *gwneud* in a northern variety of colloquial Welsh.

	future	past
1S	*wna*	*wnes*
2S	*wnei*	*wnest*
3S	*wneith*	*wnaeth*
1P	*wnawn*	*wnaethon*
2P	*wnewch*	*wnaethoch*
3P	*wnân*	*wnaethon*

Tables 2.1 and 2.2 illustrate northern colloquial Welsh paradigms of *bod* 'to be' and *gwneud* 'to do' respectively. Note that the different forms *ydy/mae* and *ydyn/maen* are due to mood, and not due to tense [3]. The other auxiliary discussed by Borsley et al. (2007), *ddaru*, does in colloquial Welsh not have any other paradigm than its past tense "ddaru" itself and accordingly is used solely as a past tense marker (and usually transcribed as such), it also does not show agreement for number and appears to be confined to northern varieties of Welsh (Borsley et al., 2007, p. 38).

Auxiliary-initial clauses then can be formed with any of these three auxiliaries, following their respective paradigms and restrictions. Examples are given in (2–1) to (2–3) below:

(2–1) *Dw* *i* *'n* *licio* *chwarae* *sboncen.*
be.1S.PRES I IMPFV like play squash
'I like playing squash.'

be expressed using the perfective statement "Dw i wedi bwyta", in all three cases "dw" is the first person present tense inflection of *bod* 'to be'. For a brief discussion of tense in colloquial Welsh see Borsley et al. (2007, pp. 9–10).

[3]Similarly, forms of *bod* beginning in a vowel may be prefixed with *d-* to mark them for negative mood and *r-* to mark them for affirmative mood, *mae/maen* are the affirmative mood equivalents of *ydy/ydyn*.

(2–2) *Wneith y parot dweud 'ta ra' yn fuan.*
do.3S.FUT the parrot say bye bye PRT soon
'The parrot will say "bye bye" soon.'

(2–3) *Ddaru Rhian sgwennu 'r aseiniad ddoe.*
PAST Rhian write the assignment yesterday
'Rhian wrote the assignment yesterday.'

As can be seen, the *bod*-derived example in (2–1) includes the imperfective aspect marker *yn*, while neither the *gwneud*-clause in (2–2) nor the *ddaru*-clause in (2–3) include such a particle. It can also be seen how the following verbs, *chwarae*, *dweud* and *sgwennu* are all present in their infinitive form, tense having been marked by the initial auxiliary.

Simple interrogatives using periphrastic constructions have the same surface structure, but differ mainly in intonation, while wh-questions make use of preposed question particles. Similarly, almost any item in such a clause can be fronted in order to focus it. An example of a simple interrogative is given in (2–4), an example of a wh-question is presented in (2–5) and (2–6) shows a construction with a focused subject.

(2–4) *Wyt ti 'n licio chwarae sboncen?*
be.2S.PRES you IMPFV like play squash
'Do you like playing squash?'

(2–5) *Be' wyt ti 'n neud?*
What be.2S you IMPFV do
'What are you doing?'

(2–6) *Rhian ddaru sgwennu 'r aseiniad.*
Rhian.FOC PAST write the assignment
'It was Rhian who wrote the assignment.'

Further, with negative statements there is an additional intervening negation particle *ddim* present, which follows the subject and in case of aspectual clauses precedes the aspect marker, as is illustrated in (2–7) and (2–8) below:

(2–7) *Wna* *i* *ddim* *dy* *helpu.*
do.1S.FUT I NEG 2S help
'I won't help you.'

(2–8) *Dydy* *o* *ddim* *yn* *gweithio* *ddoe.*
be.3S.NEG he NEG IMPFV work yesterday
'He didn't work yesterday.'

In this section I have illustrated some of the basic characteristics of auxiliaries in Welsh and how they are used within sentences. In the next section I will discuss what auxiliary deletion is and how this has been previously described in the literature.

2.3 Previous Descriptions of Auxiliary Deletion in Welsh

Auxiliary deletion as a specific point of focus has so far been discussed very little, and Davies (2010) is the only source specifically targeting the phenomenon, though Jones (2004) and Phillips (2007) also provide partial descriptions of the phenomenon. While Jones (2004) focuses on a general description in which he limits himself to discussing as factors geographic distribution and grammatical person, Davies (2010) and Phillips (2007) describe actual data containing AD which they have collected. Davies (2010,

p. 264) describes AD simply as sentences without an overt initial finite verb, where he interprets this to have been replaced by a null form of the appropriate auxiliary. He refers to these sentences as –A, while he refers to sentences with overt auxiliaries as +A. Jones (2004) does not give a specific name to the phenomenon but notes that for some speakers, depending on their regional variety, some auxiliaries, depending on grammatical number, can be omitted. An example of AD from Davies (2010) is given in 2–9 below:

(2–9) *ti* *ddim* *yn* *licio* *dreifio?*
you NEG IMPFV like drive
'You don't like driving?' (Davies, 2010, p. 267, my gloss)

From this the relation to clauses such as 2–1 can easily be seen, the only element missing being the auxiliary. It can also be assumed that this auxiliary would be one derived from *bod*, as the imperfective particle clearly indicates that this is an aspectual clause. In fact, most of the AD examples given by Davies (2010) have such an aspectual particle, though some clauses appear to also show what Davies (2010, pp. 316–322) calls particle deletion (PD), where the aspectual particle is deleted. This may happen either in conjunction with AD or not, but where it does, it of course poses a problem in inferring which auxiliary might have been deleted, and Davies (2010, p. 320) shows that a large proportion of PD clauses also feature AD (nearly 90%, though the reverse is not true).

Because of this presence mainly in aspectual clauses, Davies (2010) proposes that these sentences essentially feature a null variant of the *bod*-auxiliary, which are moreover by default interpreted as present tense. This

raises the question in the first instance, whether AD is restricted to *bod* or can occur with other forms of auxiliaries too. Davies' (2010) study also looks only at the second person singular pronoun *ti* in conjunction with AD, so that it remains open whether his findings also hold for different grammatical persons, as for instance those proposed by Jones (2004) – an issue which will be further discussed in section 2.4. However, Davies (2010) also cites some data from Roberts (1988), which contains a wh-questions with wh-fronting and Davies (2010) himself finds a construction where AD follows a conjunction. Both examples are given in 2–10 and 2–11 below:

(2–10) *lle* *ti* *'n* *mynd* *i?*
where you IMPFV go to
'Where are you going to?' (from Roberts 1988, p. 199, reported in Davies 2010, p. 276, my gloss)

(2–11) *ond* *ti* *ddim* *wedi* *sylwi* *hynny*
but you NEG PFV realise that
'But you haven't realised that.' (Davies, 2010, p. 288, my gloss)

These show that AD does also appear to not be restricted just to simple auxiliary initial clauses, but also occurs in other constructions including other overt material before them or even involving movement of a constituent around them such as in 2–10.

Davies (2010) also shows that AD appears to be the norm in current spoken colloquial Welsh, as is represented by the *Siarad* corpus. He found that 92.75% of clauses analysed featured AD (Davies, 2010, p. 285). He also gives further analyses and breakdowns for individual speakers, age groups, geographical distribution and type of construction. From this it should mainly be noted that every speaker he analysed showed AD, with 9 doing

so 100% of the time, a further 15 at least 83.33% of the time and only four speakers below that, with a single minimum of 50% (Davies, 2010, pp. 291–292). His analysis by age shows that there is statistically significant age variation, with older speakers appearing to produce slightly less AD than other groups, but the whole range of variability is only about 10% (Davies, 2010, p. 297). Regarding regional origin of speakers, he concluded that his data indicated this had "no effect on frequency of [AD]" (Davies, 2010, p. 295). Taken together this shows that AD as a phenomenon itself is both widely distributed and highly common across the speaker population and should be seen as an essential part of the language's present grammar.

While these corpus studies can show that AD is a wide-ranging phenomenon, which appears to occur in many different situations and with all speakers of modern colloquial Welsh, neither Davies (2010) nor Phillips (2007) make any specific claims about purported constraints on AD. However, some other linguists have given such constraints at least for grammatical person, presumably based on their own observations and intuitions, which I will discuss in the next section.

2.4 Auxiliary Deletion and Grammatical Person

Both Jones (2004) and Borsley et al. (2007) make brief mentions of AD and in doing so mainly focus on describing with which pronouns this occurs. As such Borsley et al. (2007, p. 260) describes the phenomenon as follows: "A

further notable feature of *bod* is that finite forms are sometimes omitted in clause-initial position in colloquial Welsh with certain pronominal subjects." This statement actually contains several constraints on AD: First, it may only occur in sentences that feature a clause-initial finite form of *bod*, i.e. with what was described as aspectual clauses in section 2.2; secondly, AD can only occur with pronominal subjects, i.e. not with proper names or "things"; and thirdly, this only applies to a subset of pronominal subjects, i.e. there is a restriction in the grammatical person and number of the pronominal subject the auxiliary agrees with.

Borsley et al. (2007) qualify their restriction to *bod* by establishing that these sentences may contain a *bod*-derived auxiliary in tag-questions, regardless of whether there was an overt auxiliary or not in the main clause. To exemplify this, Borsley et al. (2007) give the two examples repeated as 2–12 a and 2–12 b below:

(2–12) (a) *Rwyt* *ti* *'n* *mynd,* *ynd* *wyt?*
be.PRES.2S you IMPFV go Q.NEG be.PRES.2S
'You are going, aren't you?' (Borsley et al., 2007, p. 261, my gloss)

(b) *Ti* *'n* *mynd,* *ynd* *wyt?*
you IMPFV go Q.NEG be.PRES.2S
'You are going, aren't you?' (Borsley et al., 2007, p. 261, my gloss)

They demonstrate by this analogous construction that an initial auxiliary can be assumed, as with the tests on auxiliary status they discuss in Borsley et al. (2007, pp. 44–47). Additionally this supports their assumption that

clauses such as (2–12 b) are not actually without an auxiliary but rather that this is a null form of *bod*.

Regarding the restriction on pronominal subjects, Borsley et al. (2007) provide little further discussion, but they state that this "omission is particularly common with *ti* 'you.S' but it also occurs with *ni* 'we' and *chi* 'you.PL' [... and] with *fi* 'I' and *nhw* 'they' in the speech of some speakers of southern dialects" (Borsley et al., 2007, p. 261). However, this is discussed a little more at length in Jones (2004), who initially notes it as a specific feature of *wyt* that it can also be omitted but then notes that this would also work with the second person plural inflection (e.g. *dach*) and (at least in southern dialects) the first person plural inflection (e.g. *maen/ydyn*) (Jones, 2004, p. 101). He then goes on to state that in actuality, some speakers even show AD with the first person singular, where they would however use the form *fi* rather than the usual clitic *i* (Jones, 2004, p. 101). Note also that Jones (2004) makes no explicit mention of a condition which requires deleted auxiliaries to occur with pronominal subjects, though he notes that the form of the verb permitted to undergo AD is that in the copular construction, which potentially limits the scope of AD to omission of forms of *bod* 'to be' and not the general Welsh auxiliary. When it is assumed that the verb-noun in auxiliary clauses in Welsh forms the predicate of the sentence which is linked to by the auxiliary however, this would be the case with any type of the auxiliary clauses discussed in 2.2, not only in cases where there is no additional non-finite verb present (which is the case with the examples given by Jones, 2004).

Table 2.3: Comparison of AD grammaticality by grammatical person between Borsley et al. (2007) and Jones (2004).

	Borsley et al. (2007)	Jones (2004)
1S	limited	limited
2S	yes	yes
3S	no	no
1P	yes	limited
2P	yes	yes
3P	limited	no

Table 2.3 shows a summary and comparison of the conditions given by Borsley et al. (2007) and Jones (2004). From this it can be seen that while they largely agree on which grammatical person AD can occur with, they make different predictions for 1P and 3P, where Jones (2004) seems to apply greater restriction in that where Borsley et al. (2007) say that AD with 1P is acceptable, Jones (2004) says this is only the case for some speakers, and where Borsley et al. (2007) say that AD is acceptable with 3P only for some speakers, Jones (2004) does not state that this is the case, leaving the assumption that this would from his account be unacceptable. This has also been assumed with 3S in both cases, which is not mentioned in either accounts as permissible. It should also be noted that 2S and 2P are then the only cases where both accounts would lead to the prediction that these are generally acceptable.

Despite the question over which account, if either, makes more accurate predictions in an actual experimental study, Jones (2004) also makes no mention of a condition whereby AD would be grammatical only together

with a pronominal subject. However, Welsh finite verbs do not show agreement in person with non-pronominal subjects but instead default to the third person inflection of the finite verb, so that for instance in both "the child is" (singular non-pronominal subject) and "the children are" (plural non-pronominal subject) the form of *bod* 'to be' used would be the third person inflection, e.g. *mae* in a positive statement. Since both Borsley et al. (2007) and Jones (2004) have ruled out AD with 3S, one could then predict that AD with non-pronominal subjects is equally unacceptable. Jones' (2004) version does however not make explicit any requirement for pronominal subjects, so that, should this prediction of 3S's unacceptability not show to be borne out, and if he is otherwise correct, non-pronominal subjects may actually also be acceptable[4].

In the previous section it has been shown that AD is not only restricted to strict surface V1 position of the auxiliary that is deleted, but may also be possible in conjunction with wh-movement and conjunctions. In this section it was shown that some constraints on AD have however been posited in regards to what type of subject it may occur with and also which grammatical person it may occur with. In the next section, I will give a short summary of the literature that was discussed in this chapter and the implications that follow for the studies presented in this dissertation.

[4]Note that Davies (2010) does in fact find some AD clauses with non-pronominal subjects, such as "Lily'n byw 'da Kristen" (Davies, 2010, p. 270).

2.5 Summary

In this chapter I have first given a brief outline of auxiliaries and auxiliary clauses in colloquial Welsh. In this it was illustrated how auxiliaries can be derived from different verb stems and how those derived from *bod* feature additional aspectual content. It was also noted that these constructions are fairly flexible and as such while their default word order is AuxSVO, most constituents may be fronted for focus.

Section 2.3 then gave a broader outline and example of AD in Welsh and discussed some corpus based studies of AD which demonstrated how widely spread the phenomenon appears to be and that at least for the second person singular it was so common that it can quite possibly even be considered unmarked in comparison to its overt counterpart. It was also argued that the data presented through these studies shows that AD appears to occur together with some other basic configurations where the auxiliary is not the first overt item, such as with wh-fronting and after conjunctions.

In the following section, 2.4, it was then shown that some other linguists have in their outlines of AD posited different constraints to do with the type of subject AD may occur with, principally pronominal subjects, and the grammatical person of the auxiliary/subject present. It was mainly concluded that AD should be highly acceptable with 2S and 2P, not with 3S and that it is of questionable status with other grammatical persons, depending on the account followed.

Following these descriptions of AD, it is clear that grammatical con-

straints on AD have not been empirically investigated and all that is present are descriptive corpus studies giving positive examples of AD and short mentions of AD in other places which posit some constraints but are probably based on the author's intuitions and are not in agreement across authors, so that for a better understanding of the phenomenon, grammatical constraints on AD need to be further investigated.

In relating these constraints with corpus studies such as those conducted by Davies (2010) and Phillips (2007) however, it must be noted that corpus studies are unlikely to shed much light on actual grammatical constraints on the use of AD. In the case of deciding which auxiliary may have been deleted, this is simply because of the complication with AD discussed in section 2.3, which at least for clear present tense interpretable sentences does not offer any reliable source for determining which auxiliary would have been used had it been overt. It also has to be noted that while corpus studies can clearly show that some constructions are used regularly and should thus be considered part of the grammar, it is as a method never able to exclude constructions from the grammar of its language as it will never contain negative examples. This is in analogy to some of the consequences of Zipf's law, which states that there is a linear distribution of word frequencies for all the words in a given corpus along a constant defined by the relationship of a word's frequency and it's ranking in relation to other frequencies in the corpus. This leads to the prediction that the majority of words in a corpus are relatively rare or infrequent, with some occuring only once (Manning and Schütze, 1999, pp. 23–29). These non-recurring

words are known as hapax legomena[5] and when it is assumed that Zipf's law holds not only for words but also collocations of these words, it is predicted that there will also be constructions which only occur marginally in a corpus or not at all: if there are predicted to be hapax legomena for lexical items, it can be assumed that there are also hapax legomena on the basis of constructions. Occurrences of any type of construction in a corpus can then only be taken as a positive indicator for the grammatical possibility of the construction, but never as a defining corpus of all positive examples of their range[6]. It is for this reason that in order to determine the boundaries of acceptability surrounding a phenomenon such as AD, the collection of acceptability judgements, which is further discussed in chapter 4 where I conduct such a judgement experiment, appears to be one of the most appropriate methods for establishing further constraints on AD in Welsh.

[5]Crystal (2008, p. 224) gives as a definition for *hapax legomenon* "a word which occurs only once in a text, author, or extant corpus of a language". This concept is here extended beyond the word level to that of an entire construction that only occurs once in a corpus of text.

[6]This is similar to the argument of the poverty of the stimulus, where in the absence of negative evidence the insufficient range of positive evidence presents the problem for the grammatical acquisition process (cf. Chomsky, 1988).

3 Auxiliary Deletion in the *Siarad* Corpus

3.1 Introduction

As has been outlined in section 2.4 above, both Jones (2004) and Borsley et al. (2007) state that the range of pronouns AD occurs with is limited by the grammatical person of their accompanying pronouns. However, they make slightly different predictions as to which grammatical persons these would be, as has been shown in table 2.3, which compares both their predictions. Notably they differed in that Borsley et al. (2007) states that AD occurs with the first person plural and in a limited range also with the third person plural, while Jones (2004) states that occurence with the first person pluarl is limited and that AD does not occur with the third person plural. They both agree that AD is limited with the first person singular and does not occur with third person singular, while it does occur with all other pronouns.

In this chapter I describe a corpus analysis that was carried out to validify these predictions, and to see which of the two predictions, if any, reflects

the data in the corpus more accurately. For this an automated analysis of the *Siarad* corpus (Deuchar et al., 2009), a 40-hour conversational corpus of colloquial Welsh, was carried out, in which examples of sentences featuring AD were extracted. How this was achieved is further described in section 3.2 below, and section 3.3 presents and discusses the results of this corpus study, while I briefly summarise them in section 3.4.

3.2 Methodology

In order to extract instances of AD from the *Siarad* corpus, the corpus was first glossed using the Bangor AutoGlosser (Donnelly and Deuchar, 2011), a constraint-grammar based program that provides automatic tagging for corpora in the CHAT file format (MacWhinney, 2000). The version of the AutoGlosser used was cloned via *Git*[7] from its official repository[8] on Monday, 27th February 2012. Some minor changes and bug fixes[9] were made to the AutoGlosser's source code in order to make it run on a Microsoft Windows NT platform with PHP Version 5.3. While these changes should not alter the behaviour of the AutoGlosser, the author plans on committing the changes to the repository and so these should be available in later versions of the AutoGlosser, provided they are to be accepted by the repository's maintainer. Although the *Siarad* corpus was originally glossed manually, using the additional glossing tier produced by the AutoGlosser

[7] A version control system; see http://git-scm.com/ for more information.

[8] http://thinkopen.co.uk/git/autoglosser

[9] These were instances where files were linked in a Unix file hierarchy which were changed to their Windows equivalents and a number of instances where the source code resulted in E_NOTICE and E_WARNING php errors due to the use of outdated syntax or missing declarations for variables and indices.

was advantageous because the glosses it provides are more consistent, richer in detail and also because it does not gloss items that are structurally irrelevant for the purpose of this analysis, such as filled pauses, thus providing a tier for analysis that only contains the constituents that are indeed relevant to identifying AD. In order to gloss the entire corpus, the script given in program listing 1 (appendix A) was used, which runs the AutoGlosser on every CHAT file in the corpus and then extracts the glossed CHAT files from the AutoGlosser's output.

Following this, a program was written to extract instances of AD from this autoglossed corpus, the program code of which is given in program listing 2 (appendix A). This program includes a parser for the CHAT file format (lines 449 and following), which reads a CHAT file into an object representation (referred to as a *ChatDocument*) and also creates dependent objects for all the tiers and lines in the ChatDocument (which in turn are represented by the *ChatLine* classes) as well as the header data such as speaker information. With this the ChatDocument has a complete, hierarchical representation of the CHAT file it is associated with in which every line is associated with a parent object up to the ChatDocument itself. In case of the dependent tier *%aut* created by the AutoGlosser this was especially useful since it allows to trace back to the original input line that was glossed from and in turn from this the associated speaker data provided in the files headers. This was used to not only count relevant structures but extract information and the relevant data from the corpus about each instance of AD. In order to extract instances of AD, the program contains the function *find_ad()* (lines 387 and following). This function reads a given

file into a ChatDocument object and then goes through all the dependent tiers created by the AutoGlosser. Each of these lines was tested on the first item that was overtly glossed (i.e. items the AutoGlosser did not gloss but marked instead as empty, such as filled pauses, were ignored). The condition applied to this item was that it be a pronoun and also that it is only marked for number and person, i.e. it should match the regular expression `/PRON\.[0-9][S|P]/`[10]. If these conditions are met for the line under observation, this is then counted as an instance of AD and the function then collectes and returns data about it, such as the speaker, what pronoun was used, which file and where within it it occurred as well as an extract of 50 characters from the beginning of the parent ChatLine object (i.e. the line that the dependent AutoGlosser tier belongs to). The procedural part of the program ran this function and another function *extract_speaker_data()* (lines 319 and following) on every CHAT file in the autoglossed corpus. The data thus collected by the program is then written into a relational SQLite 3 database as well as three tabulator separated CSV files. The database can after easily be used to analyse the data and extract subsets depending on several conditions and relations, while the CSV files offer an easy way of importing the same set of data into spreadsheet and statistical software such as Microsoft Excel or SPSS. In terms of content both output formats are equal. The database contains three tables, *files*, *speakers*, and *ad_instances*, which are equvivalent to the similarly named CSV files. The

[10]Note that the code does not actually use the perl compatible regular expression module (PREG) supplied with PHP, but instead chunks the item and tests for these conditions to match. This approach was adopted purely for performance reasons and is otherwise equivalent to the given expression.

files table contains the filename of the CHAT file and a unique ID for this, which is referenced in the *speakers* and *ad_instances*. The *speakers* table contains all the information extracted about speakers from the CHAT file, such as their age, sex, role, etc. and also gives every speaker a unique ID per file (since the same name could plausibly occur in several files in the corpus but not necessarily refer to the same speaker), which is used for reference in the *ad_instances* table. The *ad_instances* table contains all the actual instances of AD that were extracted by the program, for each giving the file ID, the speaker ID, the line number it occured at in the CHAT file, which person the pronoun had, which number the pronoun had, a combination of these two (e.g. 1S, 2P, ...) and a 50 character extract of the relevant passage in the corpus.

3.3 Results & Discussion

A total of 1662 instances of AD were extracted from the corpus, distributed over 143 out of the total number of 156 speakers in the corpus (this count excludes speakers in the corpus whose role was given as "Investigator").

Table 3.1 gives a count of the number of instances of AD that were found for each of the grammatical persons in Welsh and how many unique speakers have produced at least one of these instances of AD. This immediately shows that AD with the second person singular is the most common, with 129 speakers producing at least one such utterance. This is followed by AD with the first person singular, which though significantly lower, was still produced by 64 speakers, almost exactly half of the number of speakers

Table 3.1: Summary of instances of AD extract and how many speakers produced them by grammatical person and number.

	Instances of AD	Unique Speakers
1S	312	64
2S	1230	129
3S	0	0
1P	56	25
2P	37	27
3P	27	24
Total	1662	143

producing AD with the second person singular. The first, second and third person plurals were all relatively rare compared to this, with 25, 27 and 24 speakers producing them respectively. Notably, there was not a single instance of AD together with the third person singular in the extracted data.

This suggests that the predictions of both Borsley et al. (2007) and Jones (2004) held true in that AD with the second person singular was very common, while it was more limited with the first person singular and never occurred with the third singular. However it appears that when it comes to the plural their predictions are less accurate, where Borsley et al. (2007) suggested that AD with the first person plural is acceptable and Jones (2004) predicted that AD with the third person plural is unacceptable, the collected data would suggest that in both cases these are rather limited, and where they both predicted that AD with the second person plural would be acceptable, the data also shows this to be more likely of limited nature. It should especially be noted that this impression is confirmed when not only

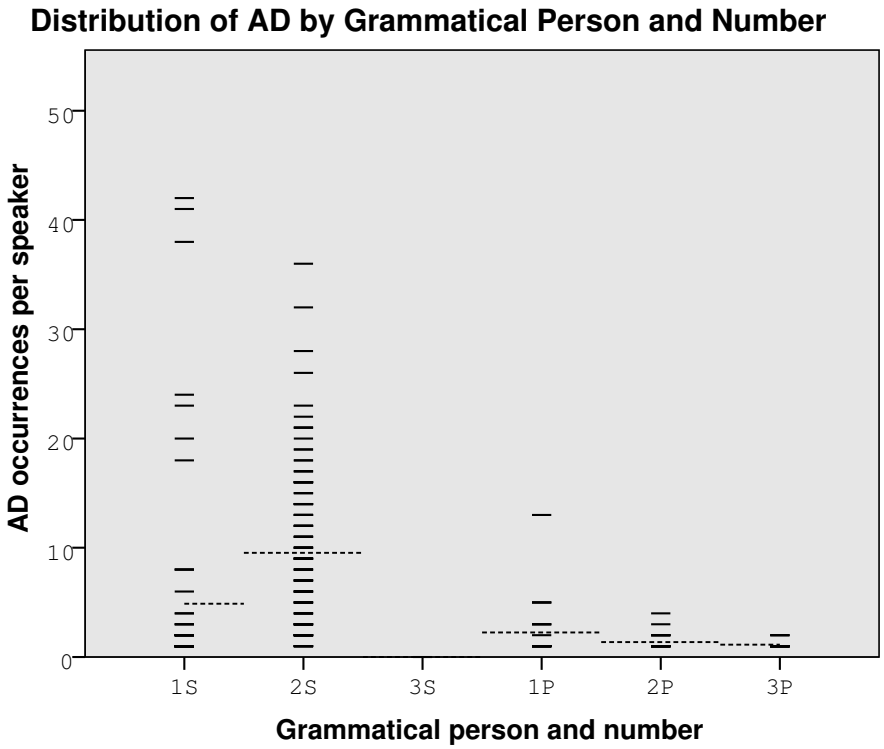

Figure 3.1: Scatter plot of the distribution of AD instances by grammatical person and number. Every small bar represents one or more speakers who produced y amount of AD instances. The dashed line shows the average number of AD instances per grammatical person and number.

the extent but also the distribution of AD instances across the speakers who produce them is taken into account. From the scatter plot in figure 3.1, which for each speaker shows how many instances of AD they produced with the given pronoun, it can be easily seen that there is very large variance in how many such utterances individual speakers produce and that this variance is greater with the more common first and second person singular. All the plural forms show very little variance, suggesting that even among

Table 3.2: Results on AD and grammatical person/number in Siarad Corpus compared to predictions from Borsley et al. (2007) and Jones (2004).

	Borsley et al. (2007)	Jones (2004)	In *Siarad* Corpus
1S	limited	limited	yes
2S	yes	yes	yes
3S	no	no	no
1P	yes	limited	limited
2P	yes	yes	limited
3P	limited	no	limited

those who produced them they may not necessarily be prevalent with the exception of one outlier in the first person plural. This speaker however was also the only child in the corpus who had a conversation with its mother, and it may be well possible that the high rate of AD with the first person plural can be attributed both to the situation and ongoing acquisition process.

This figure also gives averages (dashed lines), which are much higher in the cases of first and second person singular than in the three plural cases, where they actually approach single instances. In comparison it would then be reasonable to say that while the plural forms are limited, the first and second person singular appear to be quite widespread and common[11]. How this compares to the previous accounts from Borsley et al. (2007) and Jones (2004) is illustrated in table 3.2.

[11]Though note that this does not take into account geographical distribution, so that AD with the first person singular may indeed be more appropriately classified as limited in some parts of Wales. In terms of acceptability however one would expect that this is still given in other parts of the country when the phenomenon is so widespread.

3.4 Summary

In this chapter I have described a corpus study that looked for instances of AD in direct collocation with personal pronouns in the Welsh conversational *Siarad* corpus. I have explained how a program was used to extract these and compared the results with the accounts of AD presented by Borsley et al. (2007) and Jones (2004), showing that while they make good predictions overall, a more simple account would be that AD is common with the first and second person singular and very limited with any of the three plural pronouns, while it never occurs with the third person singular.

In the next chapter I will describe a sentence judgement experiment that was carried out after this corpus study to explore some of the wider grammatical patterns in which auxiliaries are used in Welsh and how these relate to AD. This also includes a further look at AD and grammatical person, this time not from a production point of view but explicitly from that of grammatical acceptability.

4 Sentence Judgements on Auxiliary Deletion

4.1 Introduction

In the previous chapter a corpus study was described which tested the predictions made by Jones (2004) and Borsley et al. (2007) and concluded with a slightly different set of predictions for the grammaticality of AD in relation to grammatical person and number (see table 3.2). It was noted that while a corpus study can confirm the existence of predicted constructions, it is due to its limitedness and absence of negative evidence ultimately unsuitable to identify grammatical constraints on AD.

In this chapter an acceptability judgement experiment is described which seeks to test a broad variety of the type of constructions that commonly feature auxiliary use in colloquial Welsh. From this it is hoped to gain a relatively theory-neutral overview of the acceptability of AD in a number of grammatical conditions, such as different agreement patterns, tense, mood, focus sentences, &c. The purpose of this experiment shall be to identify some of the constructions in which AD may be unacceptable. Such data it

is believed is necessary in order to be able to identify areas where further investigation is needed and also to form the basis of any theoretically motivated explanatory hypotheses for AD from an objective set of data in the future.

In order to achieve this set of acceptability judgements, a two-part experiment was constructed. With the assumption that AD is a highly verbal phenomenon not typical in written discourse (perhaps with the exception of extremely informal writing such as on social networks, e.g. *twitter* or *facebook*), it was thought that an accurate test of acceptability is best also contained in this modality, and (1997, Ch. 6) notes that mode of stimulus presentation indeed affects judgements in this way. For this reason an auditory judgement experiment was designed, in which stimuli were presented to participants over headphones who then responded on a computer keyboard. This was followed by a written task in which the same stimuli were presented as written sentences. This functioned both as a control for the auditory acceptability experiment and to gain a more finely graded response from participants through the use of Likert scales. Such a control was thought to be required not only as it will increase reliability of results through duplication of judgements, but especially because as Cowart (1997) notes, there is not much literature on the use of auditory stimuli in syntactic judgement experiments as of yet[12]. The precise procedure employed for these two tasks is further described in section 4.2 below.

[12]This can probably be attributed to the relative complicatedness of designing and executing such experiments compared with written tasks, which can for instance be distributed to a whole class attending a lecture at one time, as also noted by Cowart (1997, Ch. 6).

The constructions tested in this experiment can broadly be classified into the following six groups:

1. Subject agreement, which further tests the results already obtained through the corpus study in chapter 3, but extends this to test gender on the third person singular and non-agreeing non-pronominal subjects which default to the 3S inflection of the auxiliary discussed in section 3.4.

2. Tense and aspect, which tests some different auxiliaries that occur with different tenses and either with or without aspect, such as *bod* 'to be' plus aspectual particle, *gwneud* 'to do' without such a particle and *ddaru* 'PAST'. The auxiliary *bod* was tested for past, future and present tense with the imperfective particle *yn* and in the present tense with the perfective *wedi*. The auxiliary *gwneud* was tested for past and future tense, as there is no explicit present tense for this auxiliary, as already described in section 2.2. Notably, *ddaru* was not explicitly tested here because in an AD clause there would not be any specific surface structure that shows difference to *gwneud* sentences (they show different agreement, i.e. *ddaru* does not agree with its subject, but this is of course lost along with the auxiliary) nor does *ddaru* have any specific functionality that would prime a listener to assuming that the presented sentence is indeed a *ddaru*-clause.

3. Mood, this included the auxiliary *bod* in positive, negative and interrogative mood, which (depending on dialect) is reflected in the initial

morphology of the applicable forms of *bod*, e.g. *rwyt* 'be.PRES.POS', *dwyt* 'be.PRES.NEG' and *wyt* 'be.PRES.INT'.

4. Focus and subject/object movement. This addresses clauses in which either the subject or the object has been fronted for focus, so that differing from the default AuxSVO word order, the word orders SAuxVO, OAuxSV and VOAuxS (focused verb with pied-piped object) are tested for compatibility with AD.

5. This group aims at testing *bod*-introduced subordinate clauses, which are similar in function to English *that*-clauses. This is of interest because of the slightly different role of *bod* in highlighting clause structure and also in that *bod* here does not show inflection (though arguably it shows agreement in the form of initial consonant mutation to agree with its direct subject[13]).

6. Simple wh-questions and wh-questions involving a preposition. Wh-questions generally also show a different surface structure to the default question but are much more common than fronting for focus, so that they make for a valid separate category to be tested. In

[13]This is because the subject of *bod* in these clauses occurs as a possessive construction, which in the full form shows as a so-called sandwich construction as shown in the example below:

Roedd	*Sion*	*yn*	*meddwl*	fy	mod	i	*ar*	*y*	*tren*
be.3S.PAST	Sion	PFV	think	my	be	1S	on	the	train
i	*Loegr.*								
to	England.								

'Sion thought that I was on the train to England.'

However, in colloquial speech the possessive adjective (*fy* in the example above) is commonly dropped, so that in effect information dependent on agreement such as subject and also gender in the third person is solely indicated by the kind of mutation on *bod* in these utterances.

wh-questions with prepositions, such as "Who are you going out with?" Welsh traditionally also requires an initial preposition with pied-piped complement if the prepositional phrase is to be the element questioned, so that the surface word order is PrepOAuxSV, however Borsley et al. (2007, pp. 114–116) also discuss preposition stranding similar to English interrogatives with prepositions[14], where the preposition remains at the end of the sentence, leading to the surface structure OAuxSVPrep. Crucially Borsley et al. (2007) hypothesise that the mechanisms involved in the two types of constructions are different, so that it does not automatically follow that acceptability of AD with one of these constructions also licenses the other.

Each of these four groups was tested using a control condition with an overt auxiliary and comparing these judgements to a condition where the auxiliary was deleted. For each test case four utterances per condition were used, except in the first group, where only two utterances per condition were tested (which was thought sufficient because of the existing data on AD and subject agreement). This procedure is also described in more detail in section 4.2 below.

In this section I have described why a corpus study alone is not sufficient to obtain the data necessary to identify constraints on AD in Welsh and

[14]Besides Borsley et al. (2007), such constructions are also mentioned in Davies (2010, p. 276), who takes them from Roberts (1988), and Hendrick (1988, p. 180) who gives as his source Jones and Thomas (1977). A common attribution here seems to be that this is a recent phenomenon, where speakers apparently model the construction on the surface structure of the English equivalent. That the sources for this go back at least 25 years at the time of writing suggest however that, at least for some group of speakers, this must be a reasonably common and recurring pattern and may thus be a construction internal to these speakers' Welsh, and not just be the mirror of its English equivalent.

argued that such data can be collected via the experiment described in this chapter. In the next section I will give more detail on the methodology that was employed to do this. This is followed by section 4.3 which presents the results from the experiment and a short discussion of them, as well as section 4.4 which presents a summary of this experiment and its results.

4.2 Methodology

In the last section I have described the matter this experiment sets out to investigate and given a rough outline of the approach taken to do this, including a discussion of the kinds of stimuli and conditions that were tested in the experiment. In this section I will describe in more detail the methodology that was adopted in constructing both the auditory 'online' judgement task and the written 'offline' task which was administered to participants after completion of the online task.

Initially a list of stimuli to be used in accordance with the groups to be tested as outlined in section 4.1 was devised. This included four sentences per construction to be tested, except for the constructions in group one, which tested agreement in person and number with the direct subject, where it was thought sufficient to only test two sentences per construction due to the already existing data from the corpus study as described in chapter 3. Each sentence was then adapted to the two conditions +A (with overt auxiliary) and –A (without an over auxiliary, i.e. an AD sentence), so that all in all there were 8 stimuli per condition (4 for those in group one). Where information was lost due to AD that was important for the

meaning of the sentence, such as tense, sentences were constructed to force an interpretation in that sense. For instance when a past tense sentence was tested, a phrase such as "last year" was added to force a past tense interpretation even in the absence of the tense-carrying auxiliary. A further list of 10 stimuli were constructed for a training task (detailed below), which did not necessarily test the +A and –A conditions but also included other manipulations, which served partially to distract the participant from the +A/–A condition and also to make it easier to see from the data whether the task was understood by looking at some clearly ungrammatical stimuli in the training set. A full list of the stimuli used in the training task and in the test task is contained in appendix B, split into the same groups outlined in section 4.1. It should be noted that some items, if they fit more than one construction/group were re-used to avoid unnecessary duplication which would also highlight the condition tested to the participant. These are indexed by a reference in the form #NN to the stimuli used in their stead in the table in appendix B.

The experiment in which these stimuli were to be tested was designed to be in two parts. First, an online task in which participants would be played recorded versions of the stimuli and had to respond via key presses on a keyboard in a limited amount of time and second an offline task in which participants were given the same stimuli as a printed list but with the ability to go through them in their own time and to rate them in more detail on a Likert scale.

For the online task, all the stimuli were recorded into uncompressed 44,1kHz stereo Waveform Audio files with a Zoom H1 digital audio re-

corder and then edited in Audacity to remove any pauses before and after the stimuli. The speaker used for the recordings was a 20 year old female native speaker from east-mid Wales. In the experiment, participants were first presented with a set of instructions, which explained the procedure of the task and that they should decide whether a sentence they heard sounded natural to them or not (see appendix D for the exact text of the instructions) and that they should press the key *Z* if they felt the sentence was unnatural or *M* if they felt it sounded natural. This was followed by a practice task using the training stimuli and then the array of test stimuli (both of which were pseudo-randomised for each instantiation of the experiment) in four blocks of 38 stimuli each, with self-timed breaks in-between[15]. Presentation of each stimuli was preceded by the appearance of a fixation cross 300ms before the stimulus was played, a short beep (a 400kHz sine wave, amplitude 0.6, duration 100ms, generated with Audacity) 200ms before the stimulus was played and then a delay of 100ms immediately after which the stimuli audio file was played. This procedure was adopted to give participants both a visual and an auditory clue as to when they would hear the next sentence but not directly overlap or adjoin to that stimulus' beginning, as that was probably the most important part of most stimuli. Participants could then respond immediately from the beginning of the stimulus' playback up until 1500ms after it had completed playing. During the same time they were shown a reminder of their response options by displaying a red cross above the letter *Z* in the bottom left and a green tick with the letter *M* in the

[15]Participants were informed by an on-screen message how far they were through the task and could resume the task by pressing any key.

bottom right of the screen, reflecting the response options of *Z* for an unnatural and *M* for a natural sentence. This procedure was implemented and run using the open-source behavioural experiment software *OpenSesame* (Mathôt et al., in press; Mathôt and Theeuwes, 2011), the script for which is repeated as program code listing 3 in appendix C, on an ASUS Eee PC 1215N with Windows 7 SP1 and using Behringer HPM1000 headphones for audio playback. Participants' responses and their reaction times were logged together with the stimulus' ID (see list in appendix B) and the stimulus' playback duration, which was measured automatically using the script repeated in program code listing 4 in appendix C.

After completing the auditory online task participants were asked to complete the above described offline task. This involved a printed questionnaire containing a set of instructions followed by a list of the training stimuli and then a list of all the test stimuli, split into blocks of 30 stimuli each. Next to each of these stimuli was a Likert scale which covers the range one to five, whereas participants were told in the included set of instructions (see appendix D) that they should use the box labelled one to indicate that a sentence feels completely unnatural to them, and the box labelled five if it felt completely natural. As with the previous online task, every individual questionnaire was pseudo-randomised for each individual participant. This was achieved by generating the questionnaires automatically via the script repeated in program code listing 5, appendix C, which generates a printable HTML document. A facsimile of the one of the pages of one such document is included as a sample in appendix E. In order to be able to associate answers with their stimuli later on, a reference code

was included on every line which masked the stimulus' ID and condition behind a partially pseudo-generated number[16]. The masking was applied so that no order could be derived from these numbers by participants and a four-digit number was chosen to make it unlikely that any patterns would be visible from this. After completion by the participants, their answers were typed up manually and stored in a CSV file (cf. section 3.2). To ease this task the script in program code listing 8 was used[17].

Participants were additionally asked to complete a short questionnaire on their demographic background, which asked for their age, gender, level of education and the rough area where they grew up. Level of education was multiple-choice at either GCSEs/A-Levels, (Some) Higher Education or (Some) Postgraduate Education. The area they gave was used to later on code them to be from south, north or mid Wales, which should then serve as a rough assessor for dialectal variation[18]. Like the offline judgement questionnaire described above, this data was then typed up manually and stored in CSV files with the help of the script repeated in program code listing 7, appendix C. After all data was collected, the script repeated in program code listing 9 in appendix C was used to combine the results from the online task, the offline task and the demographic background data into a relational SQLite database, from which data can be easily extracted in different ways using the Structured Query Language (SQL) for analysis in

[16]See lines 264 and following in program code listing 5 for the precise algorithm that was used to do this.

[17]The script is intended to be run as an interactive website which displays an HTML form to insert the data and processes them before storage in a CSV file.

[18]Though note that dialectal distribution and variation in Wales is much more complex than this and so the actual place names they gave were also retained for more detailed analysis if desired.

statistical software.

Criteria for participant recruitment were that they are native Welsh speakers and that they are over 18 years of age. Ethics approval for the experiment was obtained from the College of Arts and Humanities Research Ethics Committee on 12 April 2012 and participants were subsequently recruited by advertising via bilingual posters hung up on notice boards around Bangor University, advertisement on the Bangor University online forum[19] and via the social networks *facebook* and *twitter*. A copy of the message/-poster that was used for advertising the study is included in appendix F. Additionally other native Welsh speakers known to me were e-mailed directly about the experiment and also asked to e-mail any of their friends who were native Welsh speakers. At the end of the experiment, which took on average around 30 minutes, participants were reimbursed £5 for their time. Consent was obtained prior to commencement of the experiment and participants were asked to sign the consent form replicated in appendix G.

In this section I have described in detail the methodology that was used to collect judgement data on some of the structures highlighted in section 4.1. In the next section I will present and discuss some of the results obtained from this.

4.3 Results & Discussion

In this section I will discuss the results obtained from the judgement experiment described in the previous sections. For this I will first give some

[19]http://forum.bangor.ac.uk

brief data about the participants who were recruited and the nature of the overall data collected. This will be followed by a group by group discussion of judgement results based on the six groups of constructions tested that were outlined in section 4.1, in which I will comment on any patterns visible in the data for the relevant constructions.

20 participants between the ages of 19 and 58 ($M = 31.85$) were recruited, of which 17 were from north Wales, 1 from mid Wales and 2 from south Wales. Male to female ratio was balanced at 10:10; 5 participants had GCSEs or A-levels, 8 had at least some higher education and 7 had at least some postgraduate education.

A total of 85 single judgement results from the offline task were excluded and this included all results for stimulus #8 in the test set due to a programming error which omitted these in generating the form handed out to participants. The other exclusions here were where participants missed out single questions in the offline task, which in the case of one participant resulted in 30 missing responses due to failure to complete one page of the form. In the online task, there were 176 timed out responses, though no single stimulus was affected over 4 times from this, and this was distributed over a range of 100 out of the 152 stimuli that were tested.

With concern to the overall judgements and the methodology used, one important consideration must be the reliability of the judgements obtained in the online task, as there is not much research on this method of collecting acceptability judgements. As Cowart (1997, p. 63) notes there are likely to be some differences between results for judgement experiments presented in different ways but they should still be highly correlated at large. This was

confirmed by a chi-square test on the correlation between the online task's yes/no judgements and the one-to-five scale ratings collected from the offline task for every individual participant and stimulus, which showed a highly significant relation ($\chi^2(8,\ N = 2956) = 645.59,\ p < .001$). An additional test on correlation was carried out comparing the mean online response ($1 \leq x \leq 2$) with the mean offline response ($1 \leq x \leq 5$) for every stimulus, which also showed a highly significant relationship between the two measurements ($r(150) = .79,\ p < .001$) and this relationship can also be seen together with some expected outliers in the scatter plot in figure 4.1. This shows that the data collected shows good overall reliability across the two methods of stimuli presentation used and I will on this basis propose that the differences in judgements found across the two methods are meaningful on the basis of my previous stipulation that the online judgements are closer to the criteria relevant in colloquial spoken language. In the following I will examine the results obtained on the different constructions that were tested.

The first group of constructions tested for AD acceptability with different pronominal and non-pronominal subjects. Table 4.1 contains computations of the means and standard deviations for the combined responses by construction and condition. This shows that AD is very acceptable with 2S, where it may even be preferred over an overt auxiliary by some speakers as indicated by the slightly higher mean score in the online task, though in the written task the overt auxiliary seemed to be slightly preferred. This is followed by 1P, 2P and 1S which also seem to be quite acceptable with mean ratings of acceptability well over the median of the respective scales. Notably with 3S there is a difference in acceptability between the male

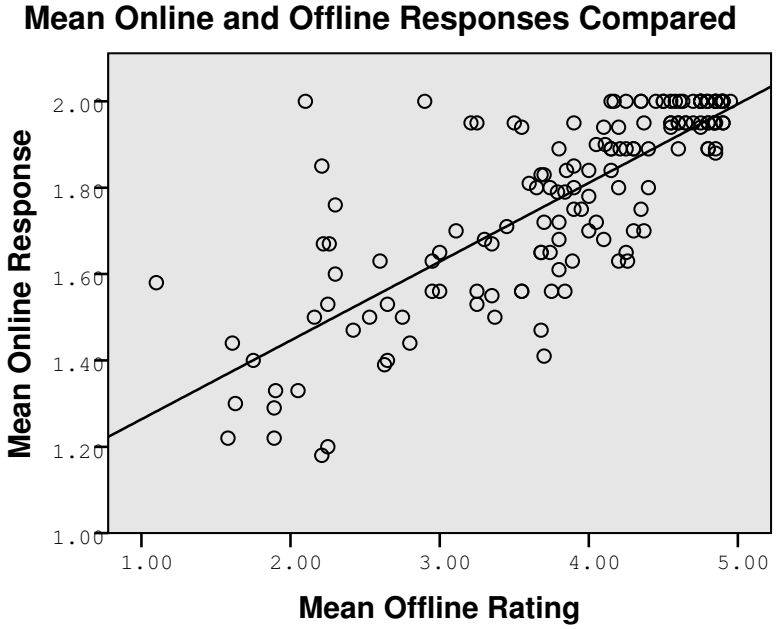

Figure 4.1: Scatter plot and trend ($R^2 = .62$) in comparison of mean responses for online and offline task per stimulus

(3Sm) and the female (3Sf) pronoun constructions, with the former being accepted much more readily in the online task, though again in the written offline task both seem to be quite unacceptable. The difference in the online responses between 3Sm and 3Sf was then shown to be significant ($t(73) = 2.76$, $p < .01$) by running an independent samples t-test. Judgements on AD sentences with noun phrases (NP) and proper nouns (PN) appear to be very unstable and varied, as indicated by their ratings close to median and standard deviations, and so possibly justify further investigation.

The second group tested some different auxiliaries (*bod* and *gwneud*) and the interaction between their tense and (with *bod*) aspect. Again means and

Table 4.1: Means and standard deviations for constructions which tested AD acceptability with different direct subjects

Subject	Condition	Online Response		Offline Response	
		Mean	SD	Mean	SD
1S	+A	2.00	.00	4.67	.70
	–A	1.68	.48	4.13	1.14
2S	+A	1.92	.28	4.17	1.36
	–A	1.97	.16	3.90	1.43
3Sf	+A	2.00	.00	4.90	.38
	–A	1.63	.49	2.26	1.29
3Sm	+A	1.95	.23	4.37	1.26
	–A	1.32	.48	2.05	1.32
1P	+A	1.97	.15	4.85	.37
	–A	1.79	.41	4.03	1.20
2P	+A	1.79	.41	4.46	.91
	–A	1.82	.39	4.05	1.08
3P	+A	2.00	.00	4.84	.44
	–A	1.46	.51	2.65	1.29
NP	+A	2.00	.00	4.90	.30
	–A	1.53	.50	2.77	1.51
PN	+A	1.90	.30	4.65	.74
	–A	1.43	.50	2.53	1.39

standard deviations were computed for every construction tested, which are given in table 4.2. In relation to tense with *bod* this shows that while the present tense constructions are highly acceptable, constructions that imply past tense are rated to be fairly unacceptable. Constructions that imply future tense appear to be more acceptable, though not as acceptable as those clearly falling within present tense. AD with *gwneud* appears to be generally relatively unacceptable, regardless of whether the sentences implied future or past tense. As the *bod.PRES* constructions were also

Table 4.2: Means and standard deviations for AD acceptability with *bod* and *gwneud*, PRES/PAST/FUT and PFV/IMPFV

Construction	Condition	Online Response		Offline Response	
		Mean	SD	Mean	SD
bod.PAST	+A	1.68	.47	3.75	1.43
	–A	1.39	.49	1.83	1.22
bod.PRES	+A	1.93	.25	4.04	1.42
	–A	1.96	.20	3.91	1.54
bod.FUT	+A	1.84	.37	4.04	1.36
	–A	1.73	.45	4.03	1.30
gwneud.PAST	+A	1.93	.25	3.81	1.46
	–A	1.30	.46	1.78	1.10
gwneud.FUT	+A	1.95	.22	4.17	1.36
	–A	1.47	.50	2.66	1.53
bod + PFV	+A	1.86	.34	4.14	1.21
	–A	1.97	.16	4.31	1.10

followed by an imperfective particle (*yn*), it is already known that this is highly acceptable at this point, and *bod* with the perfective particle (*wedi*) appears to be equally acceptable.

In the third group, the acceptability of AD in affirmative, interrogative and negative mood constructions was tested, for which means are shown in table 4.3. Here all three groups showed very high acceptability over both groups, regardless of mood, so that it can be concluded that mood is not a significant factor in AD acceptability.

The fourth group looked at AD in constructions with subject or object fronting for focus, leading to a variety of surface structures. Again a range of means and standard deviations has been computed for these constructions, which is given in table 4.4, though this does not include the

Table 4.3: Means and standard deviations for AD acceptability dependent on mood

Construction	Condition	Online Response		Offline Response	
		Mean	SD	Mean	SD
AFF	+A	1.93	.25	4.04	1.43
	–A	1.96	.20	3.91	1.54
INT	+A	1.95	.22	4.79	.47
	–A	1.97	.16	4.74	.59
NEG	+A	1.97	.16	4.59	.72
	–A	1.95	.21	4.52	.85

default AuxSVO structures, as the previous constructions in group three have already pertinently demonstrated these to be acceptable. This data shows that while all of these constructions are less acceptable overall even with an over auxiliary, those featuring AD are only slightly less acceptable than their counter-parts. Notably however, constructions with an SAuxVO surface structure[20], seemed to be much more acceptable when they were presented as spoken sentences in the online task than they were in the written offline task.

In the fifth group simple subordinate constructions roughly equatable to English *that*-clauses where tested, means and standard deviations for which are given in table 4.5. These results show that while the subordinates overall received slightly lower acceptability ratings those featuring AD are only slight lower in acceptability than those with an overt auxiliary. This is possibly indicative of AD being acceptable in these constructions, but some

[20]NB: This featured a subject in the form $[_{DP}[_{D}Y][_{N}ti]]$ 'the you', and the verb *bod* in the form *sy* rather than the 2S agreement inflection *rwyt*.

Table 4.4: Means and standard deviations for AD in different constructions with non-default surface structure

Construction	Condition	Online Response		Offline Response	
		Mean	SD	Mean	SD
SAuxVO	+A	1.87	.34	4.14	1.16
	–A	1.70	.46	2.26	1.36
VOAuxS	+A	1.66	.48	3.97	1.27
	–A	1.56	.50	3.15	1.44
OAuxSV	+A	1.61	.49	3.14	1.49
	–A	1.53	.50	3.20	1.50

Table 4.5: Means and standard deviations for AD in subordinate clauses

Construction	Condition	Online Response		Offline Response	
		Mean	SD	Mean	SD
Subordinate	+A	1.85	.36	3.90	1.27
	–A	1.71	.46	3.68	1.39

further testing would be appropriate to confirm this.

The sixth and final group tested Wh-questions, both with and without prepositions and with pied-piped and stranded prepositions. Means and standard deviations for all these three construction types are given in table 4.6. While it illustrates that constructions with stranded prepositions (Wh + Prep) are much less acceptable than those with pied-piped prepositions, there does not appear to be any significant difference between –A and +A conditions. In fact the results in the online responses are remarkably consistent across conditions. This suggests that wh-question formation does not impact on AD acceptability, and that pied-piping v preposition stranding has no effect on AD acceptability.

Table 4.6: Means and standard deviations for AD in Wh-questions

Construction	Condition	Online Response		Offline Response	
		Mean	SD	Mean	SD
Wh	+A	1.91	.29	4.41	.93
	–A	1.91	.29	4.35	1.17
Wh + Prep	+A	1.62	.49	3.74	1.23
	–A	1.63	.49	3.08	1.68
Prep + Wh	+A	1.99	.11	4.77	.48
	–A	1.95	.22	4.83	.52

4.4 Summary

In this chapter I have presented a judgement experiment in which six groups of auxiliary carrying constructions were tested for their compatibility with AD. In the previous section I have presented and discussed some of the main results from this experiment, which showed that as was already presumed from the previous studies on AD discussed and presented in this work, the type of subject to an auxiliary is of great importance in AD acceptability and notably that non-agreeing non-pronominal subjects (which fall back to the 3S inflection of the auxiliary) appear to have been judged quite inconsistently and need further investigation. It was also noted that gender in constructions with 3S appeared to be a significant factor, which has not been previously described in the literature on AD in Welsh. Tense and type of auxiliary used (e.g. *gwneud* 'to do') were also shown to be an important factor, notably AD was largely unacceptable with auxiliaries other than *bod* 'to be' and tense other than present tense, though future tense appeared to

be slightly more acceptable than past tense. On the other hand it was found that mood, aspect, movement for focus and wh-question formation had no significant impact on AD acceptability, whether they involved a pied-piping strategy or not.

In the next chapter I will briefly discuss how these findings integrate with the previous literature discussed in chapter 2 and the findings from the corpus study in presented in chapter 3 and what they implicate for further research on the constraints involved in Welsh auxiliary deletion.

5 Discussion: Possible Constraints on Auxiliary Deletion

In this dissertation I have so-far described some of the previous research that has been carried out on auxiliary deletion in Welsh and proposed that in order to gain a better understanding of the extent of the phenomenon and its implications for the grammar of Welsh, exploratory work is needed to find some of the basic constraints that apply to AD. I have followed this by describing two exploratory studies, a corpus study that looked at AD and the kinds of pronominal subjects it co-occurred with, and a judgement experiment that tested a wide array of typical Welsh constructions involving auxiliaries for their grammatical acceptability. In this chapter I will discuss what these two studies can tell us about the possible constraints that underlie AD in colloquial Welsh and how this relates to the previous literature described in chapter 2.

One of the central issues in previous descriptions was the subject of an AD clause, specifically the grammatical number and person of pronom-

inal subjects. Different accounts were given by Borsley et al. (2007) and Jones (2004) and so one of the aims of the first study was to see which of their predictions fitted better with real data found in a corpus of colloquial Welsh. The corpus study showed that while both their predictions were quite accurate for singular pronominal subjects, with the plural pronouns there was no precise agreement with either account and it was found that AD would occur with all the plural pronouns, but very limited. Based on the contrast in number of instances between AD with 1S or 2S and with 1P, 2P or 3P, it was also proposed that AD with 1S, which both previous accounts described as limited, must probably be quite acceptable regardless of the speakers dialect[21]. This was however not confirmed in the judgement experiment, which suggested that while it was still acceptable with most northern speakers, it was significantly less acceptable than 2S and 1P and 2P, which is actually more in agreement with the predictions made by Borsley et al. (2007) (cf. also table 3.2). An interesting additional discovery in the judgement experiment was that 3S, which was previously described as ungrammatical and also not found in the corpus analysis, was of roughly the same acceptability as 1S with a female pronoun, while only male pronouns were mainly unacceptable with AD. Noun phrases and proper nouns were also found to be mostly unacceptable as subjects of AD clauses, though they were for some speakers. This high variance and actually higher acceptability than production (corpus v judgements) may indeed be a sign of language change in progress, where it could be expected that if gram-

[21]One of the main assumptions for the limitedness of 1S is that it is found mainly in the southern dialect of Welsh.

maticality of AD widened this would first show in acceptability and were only then possibly followed by wider adoption. Interesting here would be a further corpus analysis focusing on data from very young speakers which could further confirm this[22].

Another major question was whether AD is limited to the auxiliary *bod* 'to be' or whether, as the term suggests, it also occurs with other Welsh auxiliaries such as *gwneud* 'to do' or *ddaru* 'PAST'. The judgement experiment's results suggested that deletion of *gwneud* is unacceptable, however there are some complications in concluding from this that AD is limited to *bod*. One of these is that *gwneud* and *ddaru* are both limited to tenses other than present tense, and this was also shown to be a constraint on AD in that past tense constructions were largely judged unacceptable and future tense constructions significantly less acceptable than present tense constructions. Additionally a problem in testing these (and indeed in their interpretability with AD) arises from what Davies (2010, pp. 326–323) describes as particle deletion, where the aspectual particle is omitted. This leads to ambiguity in sentences such as 5–2 which could have been derived from either 5–1 a or 5–1 b.

(5–1) (a) *Wyt* *ti* *'n* *siarad* *efo* *Sion?*
be.2S.PRES you IMPFV speak with Sion
'Were you speaking to Sion?'

(b) *Wnest* *ti* *siarad* *efo* *Sion?*
do.2S.PAST you speak with Sion
'Did you speak to Sion?'

[22]Davies (2010, pp. 295–302) already argues that the 2S pronoun itself shows higher adoption in younger speakers, which he suggest could be suggestive of language change in progress.

(5–2) *Ti siarad efo Sion?*
you talk with Sion
'Were you speaking to Sion?'
'Did you speak to Sion?'

This is of course easily resolved if AD remains limited to *bod*, but I suggest that further evidence is required to answer this question. A possible way here may be collecting acceptability judgements on items where the subject would show initial consonant mutation in a future tense *gwneud* clause but not in a similar *bod* clause, provided that particle drop can be shown to not lead to initial consonant mutation.

The acceptability judgements further showed that neither word order (cf. e.g. focus clauses or Wh-questions v the default AuxSVO) nor subordination constrain AD in any obvious way. Further neither mood nor aspect appeared to constrain AD either. However, as discussed above already AD is constrained by the types of subject it can occur with and presumably the agreement it shows with them as well as tense. A notable difference in the relation these two factors have to the auxiliary is that they affect its inflectional morphology, as opposed to mood which results in prefixation (if it is overt at all, which depends on dialect) and aspect exhibits no overt effect on the auxiliary. An argument here may be that the relation between the auxiliary and these constraining factors is stronger than that between the auxiliary and other factors. While this does not directly explain any bias for gender in the third person, I suggest that third person gender is in itself an important factor in that other relations depend on it, for instance in inhibiting different patterns of initial consonant mutation in adjectives; while

inflections for 3Sf and 3Sm are homophonous on the surface then, they may be different internally and the way in which speakers constrain AD may best be explained via the inflectional paradigm applied to the auxiliary.

6 Conclusions

This dissertation set out to explore some of the constraints that apply to auxiliary deletion in colloquial Welsh. It started from the viewpoint that AD had been studied very little in the wider context of the kind of constructions (specifically periphrastic constructions) in which it could potentially occur and that most descriptions of it until now focused on the pronouns it co-occurred with and whether it is motivated by language change due to the influence of English. The question followed what other factors may constrain the occurrence of AD, such as the type of auxiliary used, the constructions it occurred in, word order and movement, and the kinds of relations that affect the auxiliary in these sentences, such as mood and agreement.

Data was collected through two studies, a corpus analysis on the *Siarad* corpus of informal Welsh speech and a subsequent judgement experiment. This showed that while individual patterns of AD are highly variable, there are some clear factors that play a role in whether AD may occur in a sentence or not. It was argued that a common feature of factors that were found to constrain AD was that they had an important relation with the auxiliary, and usually one that is determinative for the inflectional morpho-

logy of the auxiliary. The two elements that showed to be vital in this were the subject AD occurred with and the tense of the clause. Further, these studies provided the first objective account on both the relative acceptability and spoken distribution of AD with pronominal subjects other than 2S, where it was shown that while previous predictions were generally quite good at predicting the data, slight differences are present. Additionally it was shown that there is a clear gap between the occurrence of these items in the speech of the *Siarad* corpus and the acceptability that speakers attach to them when they are exposed to these constructions, and in a minor fashion even that these depend on the modality through which they are experienced (i.e. auditory v visually).

In light of the existing debate over whether this phenomenon reflects language change due to the influence of English (e.g. Davies, 2010; Davies and Deuchar, in preparation), it was also noted that the data would support an analysis where the phenomenon was initially introduced in the realm of the 2S present tense paradigm of *bod*, which one would expect to also be most common in colloquial speech, and is now widening onto other parts across the inflectional paradigm of the auxiliary.

This work also highlighted some further areas of uncertainty that would warrant further experimental investigation, such as whether auxiliaries other than *bod* 'to be' can be deleted and given the acceptability ratings of 3Sf whether AD does really never occur with 3S in spontaneous speech.

Some limitations of the study and experiment were that they only looked at some very broad structures to identify the major factors that play a role, and further investigation in the areas shown to be relevant here may well

highlight some finer important details. The corpus study was also limited due to the corpus's focus on adult Welsh-English bilinguals. Re-running the corpus analysis on a corpus of children's speech and non Welsh-English bilingual speakers, such as those in the *Patagonia* corpus could give further insights and have implications for the analysis of AD as language change in progress.

References

ACCAC (2000). *English in the National Curriculum in Wales*. Cardiff: Qualifications, Curriculum and Assessment Authority for Wales, on behalf of the National Assembly for Wales.

Borsley, R. D., Tallerman, M. and Willis, D. (2007). *The Syntax of Welsh*. Cambridge: Cambridge University Press.

Chomsky, N. (1988). *Language and problems of knowledge*. Cambridge, MA: MIT Press.

Cowart, W. (1997). *Experimental Syntax: Applying Objective Methods to Sentence Judgements*. London: Sage Publications.

Crystal, D. (2008). *A Dictionary of Linguistics and Phonetics*. 6th Edition. London: Blackwell.

Davies, P. (2010). *Identifying word-order convergence in the speech of Welsh-English bilinguals*. PhD thesis. Bangor University.

Davies, P. and Deuchar, M. (in preparation). Auxiliary deletion in bilingual Welsh-English speech: internal change or the influence of English?

Deuchar, M., Parafita Couto, M. d. C., Stammers, J., Aveledo, F., Fusser, M., Jones, L., Donnelly, K., Diana, C., Davies, P. and Prys, M. (2009). *The Siarad Corpus*. [Welsh language conversational corpus]. Available: `http://siarad.org.uk`

DfES (2008). *English in the National Curriculum for Wales: Key Stages 3–4*. Cardiff: Department for Children, Education, Lifelong Learning and Skills, Welsh Assembly Government.

Donnelly, K. and Deuchar, M. (2011). *The Bangor Autoglosser: a multilingual tagger for conversational text*. [Paper presented at ITA11]. Available: `http://siarad.org.uk/publications/Donnelly2011_Bangor_Autoglosser.pdf`

Hendrick, R. (1988). *Anaphora in Celtic and Universal Grammar*. Dordrecht: Kluwer Academic.

Jones, B. M. (2004). The licensing powers of mood and negation in spoken Welsh: Full and contracted forms of the present tense of bod 'be'. *Journal of Celtic Linguistics* **8**: 87–107.

Jones, M. and Thomas, A. R. (1977). *The Welsh Language: Studies in its Syntax and Semantics*. Cardiff: University of Wales Press.

King, G. (1996). *Modern Welsh: a comprehensive grammar*. London: Routledge.

MacWhinney, B. (2000). *The CHILDES Project: Tools for Analyzing Talk*. 3rd Edition. Mahwah, NJ: Lawrence Erlbaum Associates.

Manning, C. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

Mathôt, S., Schreij, D. and Theeuwes, J. (in press). OpenSesame: An open-source, graphical experiment builder for the social sciences. *Behavior Research Methods* .

Mathôt, S. and Theeuwes, J. (2011). *OpenSesame*. [Computer Software and Manual]. Version 0.21.
Available: `http://www.cogsci.nl/opensesame` [Accessed: 2011-02-23]

Phillips, J. D. (2007). Mae nodweddion hynotaf y gymraeg ar ddiflannu. *Journal of the Literary Society of Yamaguchi University/Yamaguchi Daigaku Bungakukaishi* **57**: 261–282.

Roberts, A. E. (1988). Age-related variation in the Welsh dialect of Pwllheli. In: M. J. Ball (Ed.), *The use of Welsh: A contribution to sociolinguistics*. Clevedon: Multilingual Matters. pp. 104–122.

Appendix

A Program Code Listings for Corpus Study

Listing 1: Script for Autoglossing entire Siarad corpus

```
<?php
/***
 * Run AutoGlosser on entire corpus
 *
 * This script provides a shorthand to running the Bangor AutoGlosser on the
 * entiretiy of a given CHILDES corpus. It will assume that the AutoGlosser is
 * installed in the same directory as the script and then extract all CHAT files
 * and run the through the AutoGlosser. It will subsequently copy all the
 * relevant files to a different directory, so that this mirrors the original
 * collection of CHAT files, without the surplus output of the AutoGlosser.
 *
 * It should be used from the command line as follows:
 *   php do_directory.php <path>
 * where <path> is a required argument giving the path of the directory which
 * contains the corpus' CHAT files.
 *
 * PHP Version 5.3
 *
 * LICENSE: This piece of software was developed as part of a BA (Hons)
 * dissertation at Bangor University. It may be freely distributed and used by
 * anybody whomsoever, so long as the author is acknowledged annd no changes
 * are made to the source code without prior agreement with the author.
 *
 * @author      Florian Breit <f.breit@univ.bangor.ac.uk>
 * @copyright   2012 Florian Breit
 * @version     1.0.0
 */

//Set up PHP to report all errors
error_reporting(E_ALL);
ini_set("display_errors", 1);
ini_set("log_errors", 1);
ini_set("error_log", "./errors.log");

//Check user arguments...
```

```
36 if($argc != 2) {
37     die("This script takes exactly one argument (the path to the directory to "
38         ."be autoglossed). ".(--$argc)." arguments given.");
39 }
40
41 //Read directory and run AutoGlosser on it's CHAT files..
42 $dirname = $argv[1];
43 $dir = @dir($dirname) or die("The specified directory could not be found.");
44 @mkdir('outputs/'.basename($dir->path).'_autoglossed');
45 while(false !== $file = $dir->read()) {
46     if(substr($file, -4) == ".cha") { //Only chat files
47         $exec_file = $dir->path."/".$file;
48         //Now run do_everything for each..
49         print "\n***\n* Autoglossing file: $file\n***\n";
50         passthru("php do_everything.php \"$exec_file\"");
51         copy("outputs/".basename($file, ".cha")."/".basename($file, ".cha")
52             ."_autoglossed.txt",
53             "outputs/".basename($dir->path)."_autoglossed/".$file);
54     }
55 }
56 ?>
```

Listing 2: Script for finding AD in autoglossed corpus

```
1 <?php
2 /***
3  * Find Auxiliary Deletion in AutoGlosser data
4  *
5  * This script parses the data generated by the Bangor Autoglosser to detect any
6  * utterances which feature auxiliary deletion and generates a report for import
7  * into spreadsheet or statistical software from this.
8  * The script parses the %aut dependent tier in CHAT files generated by the
9  * Bangor AutoGlosser (http://www.siarad.org.uk/) for the first overt item, and
10 * if this is a pronoun, subject to a few other checks assumes this is an
11 * instance of AD. It compiles a list of all such instances which is then
12 * written into a SQLite3 database and also exported as the tab-separated CSV
13 * file "ad_list.csv". It also parses the original CHAT files for information
14 * about the speakers, which is then written as the the CSV file
15 * "speaker_data.csv" alongside the file index "file_list.csv". These files can
16 * then be imported into spreadsheet or statistical software, whilst the
17 * database "ad_data.sqlite" can be used for extraction of further information.
18 * The script was developed to work with the Bangor Siarad corpus, but should
19 * also work on other CHILDES corpora such as the Bangor Patagonia corpus.
20 *
21 * PHP Version 5.3
22 *
23 * LICENSE: This piece of software was developed as part of a BA (Hons)
24 * dissertation at Bangor University. It may be freely distributed and used by
25 * anybody whomsoever, so long as the author is acknowledged and no changes
26 * are made to the source code without prior agreement with the author.
27 *
28 * @author      Florian Breit <f.breit@univ.bangor.ac.uk>
29 * @copyright   2012 Florian Breit
30 * @version     1.0.0
31 */
32
33 //
34 // SETUP
```

```
35 //
36
37 //Some PHP stuff
38 error_reporting(E_ALL);
39 ini_set('display_errors', 1);
40 define('UTF8_BOM', chr(0xEF).chr(0xBB).chr(0xBF));
41
42 //Where to find the chat files for analysis
43 $original_dir = "./Siarad";
44 $autoglossed_dir = "./Siarad_autoglossed";
45 $out_dir = "./";
46
47 //
48 // MAIN SCRIPT FOR FINDING AD IN SIARAD
49 //
50
51 //Prepare database..
52 echo "Preparing database...\t\t";
53 $fh = @fopen($out_dir."/ad_data.sqlite", 'w'); //This will "empty" the db..
54 if( $fh === false ) {
55     die("\nError: Could not open file '$out_dir/ad_data.sqlite' for writing.");
56 }
57 fclose($fh);
58 $db = new SQLite3($out_dir."/ad_data.sqlite", SQLITE3_OPEN_READWRITE);
59 $result = $db->exec("CREATE TABLE files
60                             (
61                                 f_id         INTEGER PRIMARY KEY,
62                                 f_filename   TEXT
63                             );
64                         CREATE TABLE speakers
65                             (
66                                 s_id         INTEGER PRIMARY KEY,
67                                 f_id         INTEGER,
68                                 s_name_code  TEXT,
69                                 s_name       TEXT,
70                                 s_role       TEXT,
71                                 s_language   TEXT,
72                                 s_corpus     TEXT,
73                                 s_age        TEXT,
74                                 s_sex        TEXT,
75                                 s_group      TEXT,
76                                 s_SES        TEXT,
77                                 s_education  TEXT
78                             );
79                         CREATE TABLE ad_instances
80                             (
81                                 ad_id        INTEGER PRIMARY KEY,
82                                 s_id         INTEGER,
83                                 f_id         INTEGER,
84                                 ad_line_no   INTEGER,
85                                 ad_person    INTEGER,
86                                 ad_number    TEXT,
87                                 ad_persnum   TEXT,
88                                 ad_extract   TEXT
89                             );");
90 if(!$result) {
91     die('SQL Error at line '.__LINE__.': '.$db->lastErrorMsg());
92 }
93 echo "Done\n";
```

```
94
95 //Extract all file names to search for ad...
96 echo "Creating file index...\t\t";
97 $filelist = array();
98 $dir = @dir($autoglossed_dir) or die("\nThe directory with the autoglossing "
99                                      ."data could not be found.");
100 $stmt = $db->prepare('INSERT INTO files
101                         (f_filename)
102                         VALUES
103                         (:f_filename);');
104 while(false !== $file = $dir->read()) {
105     if(substr($file, -4) == ".cha") { //Only chat files
106         $stmt->reset();
107         $stmt->bindValue(':f_filename', $file);
108         $stmt->execute();
109         $f_id = $db->lastInsertRowID();
110         $filelist[] = array($file, $f_id);
111     }
112 }
113 echo "Done.\n";
114
115 //Write file list...
116 echo "Writing file list...\t\t";
117 $fh = @fopen($out_dir."/file_list.csv", 'w');
118 if( $fh === false ) {
119     die("\nError: Could not open file '$out_dir/file_list.csv' for writing.");
120 }
121 fwrite($fh, UTF8_BOM);
122 fwrite($fh, "f_id\tf_filename\n");
123 foreach($filelist as $file) {
124     fwrite($fh, implode("\t", $file)."\n");
125 }
126 fclose($fh);
127 echo "Done.\n";
128
129 //Extract speaker data..
130 echo "Extracting speaker data...\t";
131 $last_count = 0;
132 $speaker_index = array();
133 $stmt = $db->prepare('INSERT INTO speakers
134                         (
135                             f_id,
136                             s_name_code,
137                             s_name,
138                             s_role,
139                             s_language,
140                             s_corpus,
141                             s_age,
142                             s_sex,
143                             s_group,
144                             s_SES,
145                             s_education
146                         )
147                         VALUES
148                         (
149                             :f_id,
150                             :s_name_code,
151                             :s_name,
152                             :s_role,
```

```
153                                 :s_language,
154                                 :s_corpus,
155                                 :s_age,
156                                 :s_sex,
157                                 :s_group,
158                                 :s_SES,
159                                 :s_education
160                             );');
161 for($i=0; $i<count($filelist); $i++) {
162     list($filename, $f_id) = $filelist[$i];
163     shell_del_chrs($last_count);
164     $out_str = '('.($i+1).'/'.count($filelist).')';
165     echo $out_str;
166     $last_count = strlen($out_str);
167     $speakers = extract_speaker_data($original_dir."/".$filename);
168     foreach($speakers as $speaker) {
169         $stmt->reset();
170         $stmt->bindValue(':f_id',           $f_id);
171         $stmt->bindValue(':s_name_code',    $speaker['name_code']);
172         $stmt->bindValue(':s_name',         $speaker['name']);
173         $stmt->bindValue(':s_role',         $speaker['role']);
174         $stmt->bindValue(':s_language',     $speaker['language']);
175         $stmt->bindValue(':s_corpus',       $speaker['corpus']);
176         $stmt->bindValue(':s_age',          $speaker['age']);
177         $stmt->bindValue(':s_sex',          $speaker['sex']);
178         $stmt->bindValue(':s_group',        $speaker['group']);
179         $stmt->bindValue(':s_SES',          $speaker['SES']);
180         $stmt->bindValue(':s_education',    $speaker['education']);
181         $stmt->execute();
182         $s_id = $db->lastInsertRowID();
183         $speaker_index[] = array_merge(array('s_id' => $s_id,
184                                              'f_id' => $f_id),
185                                        $speaker);
186     }
187 }
188 shell_del_chrs($last_count);
189 unset($last_count);
190 echo "Done.\n";
191
192 //Write speaker data...
193 echo "Writing speaker data...\t\t";
194 $fh = @fopen($out_dir."/speaker_data.csv", 'w');
195 if( $fh === false ) {
196     die("\nError: Could not open file '$out_dir/speaker_data.csv'"
197         ."for writing.");
198 }
199 fwrite($fh, UTF8_BOM);
200 fwrite($fh, "s_id\tf_id\ts_name_code\ts_name\ts_role\ts_language\ts_corpus\t"
201             ."s_age\ts_sex\ts_group\ts_SES\ts_education\n");
202 foreach($speaker_index as $speaker) {
203     fwrite($fh, implode("\t", $speaker)."\n");
204 }
205 fclose($fh);
206 echo "Done.\n";
207
208 //Find ad lines for every file...
209 echo "Parsing files for ad lines...\t";
210 $last_count = 0;
211 $ad_index = array();
```

```
212 $stmt1 = $db->prepare('SELECT s_id, s_name_code
213                        FROM speakers
214                        WHERE f_id = :f_id;');
215 $stmt2 = $db->prepare('INSERT INTO ad_instances
216                        (
217                            s_id,
218                            f_id,
219                            ad_line_no,
220                            ad_person,
221                            ad_number,
222                            ad_persnum,
223                            ad_extract
224                        )
225                        VALUES
226                        (
227                            :s_id,
228                            :f_id,
229                            :ad_line_no,
230                            :ad_person,
231                            :ad_number,
232                            :ad_persnum,
233                            :ad_extract
234                        );');
235 for($i=0; $i<count($filelist); $i++) {
236     list($filename, $f_id) = $filelist[$i];
237     shell_del_chrs($last_count);
238     $out_str = '('.($i+1).'/'.count($filelist).')';
239     echo $out_str;
240     $last_count = strlen($out_str);
241     $stmt1->reset();
242     $stmt1->bindValue('f_id', $f_id);
243     $results = $stmt1->execute();
244     $speakers = array();
245     while($result = $results->fetchArray()) {
246         $speakers[$result['s_name_code']] = $result['s_id'];
247     }
248     $ad_lines = find_ad($autoglossed_dir."/".$filename);
249     foreach($ad_lines as $ad_line) {
250         $stmt2->reset();
251         $stmt2->bindValue(':s_id',        $speakers[$ad_line['name_code']]);
252         $stmt2->bindValue(':f_id',        $f_id);
253         $stmt2->bindValue(':ad_line_no', $ad_line['line_no']);
254         $stmt2->bindValue(':ad_person',  (int) $ad_line['g_person']);
255         $stmt2->bindValue(':ad_number',  $ad_line['g_number']);
256         $stmt2->bindValue(':ad_persnum', $ad_line['g_persnum']);
257         $stmt2->bindValue(':ad_extract', $ad_line['extract']);
258         $stmt2->execute();
259         $ad_id = $db->lastInsertRowID();
260         $ad_index[] = array('ad_id'         => $ad_id,
261                             's_id'          => $s_id,
262                             'f_id'          => $f_id,
263                             'ad_line_no'    => $ad_line['line_no'],
264                             'ad_person'     => $ad_line['g_person'],
265                             'ad_number'     => $ad_line['g_number'],
266                             'ad_persnum'    => $ad_line['g_persnum'],
267                             'ad_extract'    => $ad_line['extract']
268                             );
269     }
270 }
```

```
271 shell_del_chrs($last_count);
272 unset($last_count);
273 echo "Done.\n";
274
275 //Write ad instances...
276 echo "Writing list of AD instances...\t";
277 $fh = @fopen($out_dir."/ad_list.csv", 'w');
278 if( $fh === false ) {
279     die("\nError: Could not open file '$out_dir/ad_list.csv' for writing.");
280 }
281 fwrite($fh, UTF8_BOM);
282 fwrite($fh, "ad_id\ts_id\tf_id\tad_line_no\tad_person\tad_number\tad_persnum\t"
283             ."ad_extract\n");
284 foreach($ad_index as $ad_line) {
285     fwrite($fh, implode("\t", $ad_line)."\n");
286 }
287 fclose($fh);
288 echo "Done.\n";
289
290 echo "Script execution is complete.\n";
291
292 //
293 // CLASSES AND FUNCTIONS
294 //
295
296 /***
297  * Delete Characters from Shell STDOUT
298  *
299  * This function overwrites the last n characters on STDOUT with whitespace and
300  * then sets the cursor to the beginning of that whitespace. This only works in
301  * a shell environment when backspaces can override the current line and does
302  * not work across linebreaks.
303  *
304  * @param int $count How many characters to overwrite
305  * @return void
306  */
307 function shell_del_chrs($count) {
308     for($i=0;$i<$count;$i++) {
309         echo chr(8); //return to left
310     }
311     for($i=0;$i<$count;$i++) {
312         echo ' '; //overwrite with ws
313     }
314     for($i=0;$i<$count;$i++) {
315         echo chr(8); //return to left
316     }
317 }
318
319 /***
320  * Extract speaker data from CHAT files
321  *
322  * This function extracts all available data about participants from the given
323  * CHAT file.
324  *
325  * @param string $filename The CHAT file from which the data should be extracted
326  * @return array Returns a numeric array of the speaker information
327  */
328 function extract_speaker_data($filename) {
329     //Open and parse file
```

```
330     $cf = new ChatDocument($filename);
331     $cf->parseFile();
332 
333     //Get all header lines
334     $speaker_data = array();
335     $header_lines = $cf->getHeaderLines();
336     foreach($header_lines as $header_line) {
337         switch(strtolower($header_line->getIdentifier())) {
338             case 'participants':
339                 //data will be of the format XXX Name Role, XXX Name Role, ...
340                 $parts_header = $header_line->getData();
341                 $parts_header = explode(',', $parts_header);
342                 foreach($parts_header as $parts_item) {
343                     $parts_item = explode(' ', trim($parts_item), 3);
344                     $id = $parts_item[0];
345                     if(count($parts_item) < 3) {
346                         //no name is given (names are optional)
347                         $name = '';
348                         $role = $parts_item[1];
349                     } else {
350                         $name = $parts_item[1];
351                         $role = $parts_item[2];
352                     }
353                     $filename = $header_line->getParent()->getFilename();
354                     $filename = basename($filename);
355                     $speaker_data[$id] = array('name_code' => $id,
356                                                'name' => $name,
357                                                'role' => $role
358                                                );
359                 }
360                 break;
361             case 'id':
362                 //Format is: lang|corpus|code|age|sex|group|SES|role|edu|
363                 //Index:      0    1     2    3   4   5     6   7
    8
364                 $id_header = $header_line->getData();
365                 $id_header = explode('|', $id_header);
366                 $speaker_data[$id_header[2]] += array('language'  => $id_header[0],
367                                                       'corpus'    => $id_header[1],
368                                                       'age'       => $id_header[3],
369                                                       'sex'       => $id_header[4],
370                                                       'group'     => $id_header[5],
371                                                       'SES'       => $id_header[6],
372                                                       'education' => $id_header[8]
373                                                       );
374                 break;
375         }
376     }
377 
378     //replace array keys with numbered index
379     $new_speaker_data = array();
380     foreach($speaker_data as $item) {
381         $new_speaker_data[] = $item;
382     }
383 
384     return $new_speaker_data;
385 }
386 
387 /***
```

```
388  * Find instances of Auxiliary Deletion in an AutoGlosser CHAT file
389  *
390  * This function searches the given CHAT file's dependent tier %aut line
391  * generated by the Bangor AutoGlosser for instances where the first over item
392  * is a personal pronoun and returns an array
393  *
394  * @param string $filename The CHAT file which should be parsed for AD instances
395  * @return array Returns an array of AD instances in the specified CHAT file
396  */
397 function find_ad($filename) {
398     //Open and parse file
399     $cf = new ChatDocument($filename);
400     $cf->parseFile();
401
402     //Get all autoglosser lines
403     $aut_lines = array();
404     $part_lines = $cf->getPartLines();
405     foreach($part_lines as $part_line) {
406         $dependent_lines = $part_line->getDependentLines();
407         foreach($dependent_lines as $dependent_line) {
408             if($dependent_line->getIdentifier() == 'aut') {
409                 $aut_lines[] = $dependent_line;
410             }
411         }
412     }
413     unset($part_line, $part_lines, $dependent_line, $dependent_lines);
414
415     //Find lines that begin with pronouns
416     $ad_lines = array();
417     foreach($aut_lines as $aut_line) {
418         $first_item = trim($aut_line->getData()); //rm any empty glosses
419         $first_item = substr($first_item, 0, strpos($first_item, ' ')); //1st ws
420         if(!empty($first_item)) {
421             $first_item = explode('.', $first_item);
422             if( count($first_item) == 3          //match for xxx.xxx.xxx
423                 && $first_item[1] == 'PRON'      //match for xxx.PRON.xxx
424                 && is_numeric($first_item[2][0])//march for xxx.xxx.(0-9)xx
425                 ) {
426                 //This is probably an AD clause! It starts with a pronoun..
427                 //Now gather data about it...
428                 $filename = $aut_line->getParent()->getParent()->getFilename();
429                 $filename = basename($filename);
430                 $line_no = $aut_line->getOrigLineNo();
431                 $speaker = $aut_line->getParent()->getIdentifier();
432                 $g_person = $first_item[2][0];
433                 $g_number = $first_item[2][1];
434                 $extract = substr($aut_line->getParent()->getData(), 0, 50);
435                 $ad_lines[] = array('name_code' => $speaker,
436                                     'g_person' => $g_person,
437                                     'g_number' => $g_number,
438                                     'g_persnum' => $g_person.$g_number,
439                                     'line_no' => $line_no,
440                                     'extract' => $extract
441                                     );
442             }
443         }
444     }
445
446     return $ad_lines;
```

```
447 }
448
449 /***
450  * Root Class for CHAT Objects
451  *
452  * This is a generic root class from which all other CHAT Objects are derived.
453  * It cannot be directly instanciated.
454  *
455  * @package ChatTools
456  * @abstract
457  */
458 abstract class ChatObject {
459
460     //dummy class
461 }
462
463 /***
464  * CHAT Document Class
465  *
466  * This class provides functionality for reading, parsing, modifying and writing
467  * CHAT files as used by in the CHILDES project.
468  * If parses the lines in the CHAT file and builds a structure from these so
469  * that every line has a parent showing its relations to other lines in the CHAT
470  * document. Headers and Participant lines are children of the ChatDocument,
471  * while the dependent tier lines are children of their headin Participant line.
472  *
473  * @package ChatTools
474  * @link http://childes.psy.cmu.edu/manuals/chat.pdf The manual for CHAT files
475  */
476 class ChatDocument extends ChatObject {
477
478     /***
479      * Filename of the CHAT file the ChatDocument operates on
480      *
481      * This should be set and retrieved using the setFilename() and
482      * getFilename() methods, which ensure that the file exists and is
483      * writeable.
484      *
485      * @access protected
486      */
487     protected $filename;
488     /***
489      * Array of the header lines in the CHAT document
490      *
491      * This is an array of all the header lines in the CHAT document. It may be
492      * retrieved or modified using the setHeaderLines() and getHeaderLines()
493      * methods.
494      *
495      * @access protected
496      */
497     protected $header_lines;
498     /***
499      * Array of the participant lines in the CHAT document
500      *
501      * This is an array of all the participant lines in the CHAT document. These
502      * have the dependend tier lines as children. It may be retrieved or
503      * modified using the setPartLines() and getPartLines() methods.
504      *
505      * @access protected
```

```
506      */
507     protected $part_lines;
508
509     /***
510      * Class Constructor
511      *
512      * This is the class constructor. It takes one argument, which is the
513      * filename of the CHAT file that shall be manipulated. If you want to
514      * create a new CHAT file, you must first create an empty file which you
515      * can then manipulate with the class. The file must exist and be writeable.
516      * Note that the class does not automatically parse the file upon creation,
517      * so if it is not a new file you must still call the parseFile() method to
518      * parse it.
519      *
520      * @param string $filename Filename of the CHAT file to be loaded
521      * @access public
522      * @return void
523      */
524     public function __construct($filename) {
525         $this->setFilename($filename);
526     }
527
528     /***
529      * Set filename of CHAT document
530      *
531      * This sets the filename of the CHAT document. It is automaticlly called
532      * when the object is created and may later be used to modify the filename,
533      * e.g. when you want to save the file under a different name after having
534      * manipulated it. The given file must both exist and be writeable, if it
535      * is intended to be a new file, you must first create it.
536      *
537      * @param string $filename The new filename to use for the document
538      * @access public
539      * @return void
540      */
541     public function setFilename($filename) {
542        $this->checkFile($filename);
543        $this->filename = $filename;
544     }
545
546     /***
547      * Get filename of CHAT document
548      *
549      * This returns the filename currently used by the ChatDocument.
550      *
551      * @return string Returns the filename of the document
552      * @access public
553      */
554     public function getFilename() {
555         return $this->filename;
556     }
557
558     /***
559      * Set CHAT Header Lines for the CHAT document
560      *
561      * This function lets you replace the complete set of header lines used by
562      * the CHAT document. It must be given as an indexed array, each item of
563      * which is a valid ChatHeaderLine object whith this instance of the
564      * ChatDocument as its parent.
```

```
565         *
566         * @param $lines The array of ChatHeaderLine objects to be used
567         * @access public
568         * @return void
569         */
570        public function setHeaderLines(array $lines) {
571            foreach($lines as $line) {
572                if( !is_a($line, 'ChatHeaderLine') ) {
573                    throw new InvalidArgumentException("The given array of ChatLine"
574                                                       ." headers contains members "
575                                                       ."that are not valid "
576                                                       ."ChatHeaderLine objects.");
577                }
578            }
579            $this->header_lines = $lines;
580        }
581
582        /**
583         * Get CHAT Header Lines for the CHAT document
584         *
585         * This returns a numeric array of all the header lines of the CHAT document
586         *
587         * @return array An array of all the headers in the document
588         * @access public
589         */
590        public function getHeaderLines() {
591            return $this->header_lines;
592        }
593
594        /**
595         * Set CHAT Participant Lines for the CHAT document
596         *
597         * This function lets you replace the complete set of participant lines used
598         * by the CHAT document. It must be given as an indexed array, each item of
599         * which is a valid ChatPartLine object whith this instance of the
600         * ChatDocument as its parent.
601         *
602         * @param $lines The array of ChatPartLine objects to be used
603         * @access public
604         * @return void
605         */
606        public function setPartLines(array $lines) {
607            foreach($lines as $line) {
608                if( !is_a($line, 'ChatPartLine') ) {
609                    throw new InvalidArgumentException("The given array of "
610                                                       ."participant ChatLines "
611                                                       ."contains members that are "
612                                                       ."not valid ChatPartLine "
613                                                       ."objects.");
614                }
615            }
616            $this->part_lines = $lines;
617        }
618
619        /**
620         * Get CHAT Participant Lines for the CHAT document
621         *
622         * This returns a numeric array of all the participant lines of the CHAT
623         * document
```

```
624         *
625         * @return array An array of all the participant lines in the document
626         * @access public
627         */
628        public function getPartLines() {
629            return $this->part_lines;
630        }
631
632        /**
633         * Check whether the specified file exists and is writeable
634         *
635         * This method checks whether the file specified by $filename exists and is
636         * writeable. The $filename argument is optional and if not given the
637         * current filename of the ChatDocument will be used instead.
638         *
639         * @param string $filename The path to the file to check
640         * @return bool Returns true if the filename is valid, otherwise throws
641         *              an InvalidArgumentException.
642         * @access protected
643         */
644        protected function checkFile($filename=null) {
645            if( $filename == null ) {
646                $filename = $this->filename;
647            }
648            if( !file_exists($filename) ) {
649                throw new InvalidArgumentException("The specified file '$filename' "
650                                                   ."does not exist.");
651            }
652            if( !is_writable($filename) ) {
653                throw new InvalidArgumentException("The specified file '$filename' "
654                                                   ."is not writeable.");
655            }
656            return true;
657        }
658
659        /**
660         * Parse the associated CHAT file
661         *
662         * This method will parse the associated CHAT file (see $filename) and
663         * overwrite any current header and participant lines with those from the
664         * file. Note that the dependent tier lines are accessible through their
665         * parent ChatPartLine objects.
666         *
667         * @return void
668         * @access public
669         */
670        public function parseFile() {
671            $this->checkFile();
672            $lines = file($this->filename);
673            $last_part_line = false;
674            for($i=0; $i<count($lines); $i++) {
675                $lines[$i] = rtrim($lines[$i], "\r\n");
676                switch($lines[$i][0]) {
677                    case '@':
678                        $x = new ChatHeaderLine($this, $lines[$i]);
679                        $x->setOrigLineNo($i+1);
680                        $this->header_lines[] = $x;
681                        break;
682                    case '*':
```

```
683                     $last_part_line = new ChatPartLine($this, $lines[$i]);
684                     $last_part_line->setOrigLineNo($i+1);
685                     $this->part_lines[] = $last_part_line;
686                     break;
687                 case '%':
688                     $x = new ChatDependentLine($last_part_line,
689                                                $lines[$i],
690                                                $last_part_line);
691                     $x->setOrigLineNo($i+1);
692                     break;
693             }
694         }
695         unset($x, $last_part_line);
696     }
697 }
698
699 /***
700  * Base class for CHAT lines
701  *
702  * This is an abstract class that provides some base functionality for all types
703  * if CHAT lines: header lines, participant lines, and dependent tier lines.
704  * These three different types of lines have their own respective classes
705  * derived from this class: ChatHeaderLine, ChatPartLine and ChatDependentLine.
706  * ChatLine cannot be used directly, but you can use it to check whether a given
707  * object is any type of CHAT line with the is_a() function.
708  *
709  * @abstract
710  */
711 abstract class ChatLine extends ChatObject {
712
713     /***
714      * Reference to Parent ChatObject
715      *
716      * This is a reference to the line's parent ChatObject. This may be another
717      * ChatLine or a ChatDocument. The parent can only be set on construction
718      * and thereafter not be modified. You can use the method getParent() to
719      * obtain a reference to the parent item of a ChatLine.
720      *
721      * @access protected
722      */
723     protected $parent;
724     /***
725      * Identifier of the CHAT line
726      *
727      * Every CHAT line has an identifier, usually inbetween one of @, * or % and
728      * a colon :, these are three letters long for participant lines and
729      * dependent tier lines and can be of varying length for header lines. A
730      * special case are the Begin and End identifiers, which are not followed by
731      * a colon in the CHAT file. Note that the identifier maintained here does
732      * not have any of the @, *, % and : characters, since they are predictable
733      * from the other properties of the object. So for "*EXA:" this would
734      * contain the string "EXA", for "@Comment:" it would be "Comment", etc.
735      * This can be modified using the setIdentifier() and getIdentifier()
736      * methods.
737      *
738      * @access protected
739      */
740     protected $identifier;
741     /***
```

```
742         * Line data of the CHAT line
743         *
744         * This contains the actual data of the given CHAT line, i.e. what normally
745         * follows the identifier and a tab character. This would be things such as
746         * the actual transcription or the gloss text, depending on the type of CHAT
747         * line.
748         * You should use the setData() and getData() methods to modify this.
749         *
750         * @access protected
751         */
752        protected $data;
753        /**
754         * Original Line Number
755         *
756         * If the line was parsed from an existing CHAT file, then this contains the
757         * line number at which the line was originally positioned in the file when
758         * parsed. This can be useful for finding it in the raw file data if needed.
759         * If the ChatLine was not originally parsed from a file this is 0.
760         * Otherwise it will be any number of 1 or above.
761         * You may optionally use the setOrigLineNo() and getOrigLineNo() methods to
762         * modify this value.
763         *
764         * @access public
765         */
766        public $orig_line_no = 0;
767
768        /**
769         * Constructor for ChatLine objects
770         *
771         * This is a generic constructor function for ChatLine objects. It takes the
772         * parent document and the raw line data (not the line data contained in the
773         * ChatLine object) as its arument.
774         *
775         * @param ChatDocument $parent The parent ChatDocument for the line
776         * @param string $data The raw, unparsed, line from the CHAT file
777         * @access public
778         * @return void
779         */
780        public function __construct(ChatDocument $parent, $data) {
781            $this->parent = $parent;
782            $this->parseLine($data);
783        }
784
785        /**
786         * Set the Original Line Number
787         *
788         * This sets the original line number of the ChatLine object. This should be
789         * a reference to where the line was originally positioned in the CHAT file
790         * before parsing.
791         *
792         * @param int $line_no The line number of the line in the CHAT file
793         * @return void
794         * @access public
795         */
796        public function setOrigLineNo($line_no) {
797            $this->orig_line_no = (int) $line_no;
798        }
799
800        /**
```

```
801         * Get the Original Line Number
802         *
803         * This returns the original line number of the ChatLine object. This is a
804         * reference to where the line was originally positioned in the CHAT file
805         * before parsing. This may be useful for looking up the raw data in the
806         * CHAT file.
807         *
808         * @return int Returns the original line number
809         * @access public
810         */
811        public function getOrigLineNo() {
812            return (int) $this->orig_line_no;
813        }
814
815        /**
816         * Parse a line from raw data
817         *
818         * This method parses the given data into the identifier and the line data
819         * and stores these in the present ChatLine object.
820         *
821         * @param $data The unparsed, raw data from the CHAT file
822         * @return void
823         * @access protected
824         */
825        protected function parseLine($data) {
826            //Identifier: *XXX: -> XXX; %xxx: -> xxx; @x...x:, -> x...x
827            $this->identifier = substr($data, 1, strpos($data, ':')-1);
828            //Remaining data on line
829            $tier = substr($data, strpos($data, "\t")+1);
830            $this->data = $tier; //Eventually one could break down the individual
831                                 //items on the tier...
832        }
833
834        /**
835         * Get Parent ChatObject
836         *
837         * This returns a reference to the parent ChatObject of the present ChatLine.
838         *
839         * @return ChatObject Returns the parent ChatObject
840         * @access public
841         */
842        public function getParent() {
843            return $this->parent;
844        }
845
846        /**
847         * Get Identifier
848         *
849         * This returns the identifier of the ChatLine object. See the description
850         * of the $identifier variable for more information on what this is.
851         *
852         * @return string Returns the identifier of the CHAT line
853         * @access public
854         */
855        public function getIdentifier() {
856            return $this->identifier;
857        }
858
859        /**
```

```
860         * Get Line Data
861         *
862         * This method returns the line data for the present ChatLine. This is
863         * usually what comes behind the line identifier (e.g. "xyz in "*EXA: xyz").
864         * See the description of the $data variable for further information.
865         *
866         * @return Returns the line data for the CHAT line
867         * @access public
868         */
869        public function getData() {
870            return $this->data;
871        }
872    }
873
874    /**
875     * CHAT Header Line Class
876     *
877     * This class implements representations of header lines (lines beginning with @
878     * in the CHAT file format). At present, it's functionality is identical to that
879     * of ChatLine and so its main use if for type hinting purposes.
880     */
881    class ChatHeaderLine extends ChatLine {
882
883        //this does not provide any extra functionality to other ChatLines
884    }
885
886    /**
887     * CHAT Participant Line Class
888     *
889     * This class implements representations of participant lines (lines that begin
890     * with an asterisk * in CHAT files). It extends the ChatLine object for some
891     * functionality relating to it's ability to have subordained dependent tier
892     * lines.
893     */
894    class ChatPartLine extends ChatLine {
895
896        /**
897         * The Line's Dependent Tier
898         *
899         * This contains an array of all the dependent tier lines which are
900         * dependent on the participant line.
901         *
902         * @access protected
903         */
904        protected $dependent_lines = array();
905
906        /**
907         * Add a dependent line
908         *
909         * This adds a dependent tier line to the participant line. If the lines is
910         * already dependent on the participant line it the method call will be
911         * ignored as references are unique.
912         *
913         * @param ChatDependentLine $line The dependent tier line to be added
914         * @return void
915         * @access public
916         */
917        public function addDependentLine(ChatDependentLine $line) {
918            if( !in_array($line, $this->dependent_lines) ) {
```

```
919             $this->dependent_lines[] = $line;
920         }
921     }
922
923     /***
924      * Set all dependent lines
925      *
926      * This method is similar to addDependentLine() but it allows for the whole
927      * array of dependent tier lines to be replaced at once.
928      *
929      * @param array $lines An array of ChatDependentLine objects
930      * @return void
931      * @access public
932      */
933     public function setDependentLines(array $lines) {
934         foreach($lines as $line) {
935             if(!is_a($line, 'ChatDependentLine')) {
936                 throw new InfiniteIterator("The given array of dependent "
937                                            ."ChatLines contains members that "
938                                            ."are not valid ChatDependentLine "
939                                            ."objects.");
940             }
941         }
942         $this->dependent_lines = $lines;
943     }
944
945     /***
946      * Get dependent lines
947      *
948      * This returns an array of all the dependent tier lines associated with
949      * this CHAT line.
950      *
951      * @return array An array of ChatDependentLine objects
952      * @access public
953      */
954     public function getDependentLines() {
955         return $this->dependent_lines;
956     }
957 }
958
959 /***
960  * CHAT Dependent Tier Line Class
961  *
962  * This class extends the ChatLine class for some changed functionality.
963  * Specifically since dependent tier lines are dependent on participant lines
964  * and not ChatDocuments, it changes this so the parent document must be a
965  * CharPartLine, not a ChatDocument.
966  */
967 class ChatDependentLine extends ChatLine {
968
969     /***
970      * ChatDependentLine constructor
971      *
972      * This is the constructor for dependent tier lines. It behaves like the
973      * constructor for ChatLine but instead of a ChatDocument for the $parent
974      * parameter it expects a CharPartLine.
975      *
976      * @param ChatPartLine $parent The parent ChatPartLine for the line
977      * @param string $data The raw, unparsed, line from the CHAT file
```

```
978         * @access public
979         * @return void
980         */
981        public function __construct(ChatPartLine $parent, $data) {
982            $this->parent = $parent;
983            $this->parent->addDependentLine($this);
984            $this->parseLine($data);
985        }
986    }
987    ?>
```

B Stimuli for Judgement Experiment

Table B.1: Training Stimuli for Judgement Experiment

#	Construction	+A	–A
1	training	*Dwyt Wyn ddim isio yfed bara brith o gwbl!	*Wyn ddim isio yfed bara brith o gwbl!
2		Be' oedd y derwydd yn licio mwyaf?	*Be' yn derwydd y licio mwyaf?
3		Pryd wyt ti'n gorffen dy arholiad?	Pryd ti'n gorffen dy arholiad?
4		Mae Sioned yn gwerthu lot o stwff yn y farchnad.	*Sioned yn gwerthu lot o stwff yn y farchnad.
5		*Wnâth hi ddim cael lle oddi wrth y stafell.	Doedd gynni hi ddim lle yn eu 'stafell.

Table B.2: Test Stimuli for Judgement Experiment

GROUP I (Grammatical Person/Number)

#	Construction	+A	–A
1	1S	Dw i'n licio hufen iâ.	Fi'n licio hufen iâ.
2		Dw i'n byw ym Mangor.	Fi'n byw ym Mangor.
3	2S	*#43*	*#43*
4		*#44*	*#44*
5	3Sf	Mae hi'n byw yng Nghaerdydd.	Hi'n byw yng Nghaerdydd.
6		Mae hi'n astudio Seicoleg yn y Brifysgol.	Hi'n astudio Seicoleg yn y Brifysgol.
7	3Sm	Mae o'n dod o Ryl yn wreiddiol.	O'n dod o Ryl yn wreiddiol.
8		Mae o'n yfed lot o gwrw.	O'n yfed lot o gwrw.

9	1P	Dan ni'n neidio o gwmpas ar y gwely.	Ni'n neidio o gwmpas ar y gwely.
10		Dan ni'n licio mynd i Sbaen.	Ni'n licio mynd i Sbaen.
11	2P	Dach chi'n dadlau â'r bobl drws nesaf bob hyn a hyn.	Chi'n dadlau â'r bobl drws nesaf bob hyn a hyn.
12		Dach chi'n byw'n bell o bob man.	Chi'n byw'n bell o bob man.
13	3P	Maen nhw'n mynd am dro ar y traeth.	Nhw'n mynd am dro ar y traeth.
14		Maen nhw'n gwisgo yr un crys-t.	Nhw'n gwisgo yr un crys-t.
15	NP	Mae'r plant yn chwarae efo ffrind.	Y plant yn chwarae efo ffrind.
16		Mae'r gath yn yfed llefrith.	Y gath yn yfed llefrith.
17	PN	Mae Sian yn licio pêl-droed yn fawr.	Sian yn licio pêl-droed yn fawr.
18		Mae Rhian yn siarad Almaeneg hefyd.	Rhian yn siarad Almaeneg hefyd.

GROUP II (Tense/Aspect)

#	Construction	+A	–A
19	bod+PAST	Roeddet ti'n siopa am oriawr dwy flynedd yn ôl.	Ti'n siopa am oriawr dwy flynedd yn ôl.
20		Roeddet ti'n ymweld â dy nain di wythnos dwytha'.	Ti'n ymweld â dy nain di wythnos dwytha'.
21		Roeddet ti'n ffonio fi neithiwr.	Ti'n ffonio fi neithiwr.
22		Roeddet ti'n canu mewn côr yn blentyn.	Ti'n canu mewn côr yn blentyn.
23	bod+PRES	#43	#43
24		#44	#44
25		#45	#45
26		#46	#46
27	bod+FUT	Byddi di'n galw dy daid di 'fory.	Ti'n galw dy daid di 'fory.

28		Byddi di'n siarad efo fy athro i wythnos nesaf.	Ti'n siarad efo fy athro i wythnos nesaf.
29		Byddi di'n chwarae pêl-fasged efo Dylan nes ymlaen.	Ti'n chwarae pêl-fasged efo Dylan nes ymlaen.
30		Byddi di'n mynd ar wyliau yn y Swistir flwyddyn nesaf.	Ti'n mynd ar wyliau yn y Swistir flwyddyn nesaf.
31	gwneud+PAST	Wnest ti neud dy waith cartref di yn dda ddoe.	Ti neud dy waith cartref di yn dda ddoe.
32		Wnest ti fwydo'r planhigion mwy nag wythnos yn ôl.	Ti fwydo'r planhigion mwy nag wythnos yn ôl.
33		Wnest ti enill y gêm tro dwytha'.	Ti enill y gêm tro dwytha'.
34		Wnest ti ateb y neges John neithiwr.	Ti ateb y neges John neithiwr.
35	gwneud+FUT	Wnei di 'sgubo 'fory.	Ti 'sgubo 'fory.
36		Wnei di drwsio'r ffenest ac wna i drwsio'r drws.	Ti drwsio'r ffenest ac wna i drwsio'r drws.
37		Wnei di baratoi'r bwyd ar gyfer y parti wythnos nesaf.	Ti baratoi'r bwyd ar gyfer y parti wythnos nesaf.
38		Wnei di nôl y plant o'r ysgol yfory.	Ti nôl y plant o'r ysgol yfory.
39	bod+PFV	Rwyt ti 'di gwastrafi amser.	Ti 'di gwastrafi amser.
40		Rwyt ti 'di golchi'r llestri.	Ti 'di golchi'r llestri.
41		Rwyt ti 'di darllen yr holl lyfr.	Ti 'di darllen yr holl lyfr.
42		Rwyt ti 'di enill y cwis tafarn.	Ti 'di enill y cwis tafarn.

GROUP III (Mood)

#	Construction	+A	–A
43	bod+AFF	Rwyt ti'n chwarae tennis yn dda.	Ti'n chwarae tennis yn dda.
44		Rwyt ti'n tecstio at dy ffrindiau yn aml.	Ti'n tecstio at dy ffrindiau yn aml.

45		Rwyt ti'n gwrando at Radio Cymru.	Ti'n gwrando at Radio Cymru.
46		Rwyt ti'n eistedd yn y 'stafell fyw.	Ti'n eistedd yn y 'stafell fyw.
47	bod+INT	Wyt ti'n bwyta cinio rwan?	Ti'n bwyta cinio rwan?
48		Wyt ti'n licio mynd am dro?	Ti'n licio mynd am dro?
49		Wyt ti'n cael cawod heno?	Ti'n cael cawod heno?
50		Wyt ti'n dilyn Pobol y Cwm ar S4C?	Ti'n dilyn Pobol y Cwm ar S4C?
51	bod+NEG	Dwyt ti ddim yn yfed lot fel arfer.	Ti ddim yn yfed lot fel arfer.
52		Dwyt ti byth yn bwyta siocled.	Ti ddim yn bwyta siocled.
53		Dwyt ti ddim yn siarad efo Glyn rhagor.	Ti ddim yn siarad efo Glyn rhagor.
54		Dwyt ti ddim yn cael mynd adra eto.	Ti ddim yn cael mynd adra eto.

GROUP IV (Focus and Subject/Object-Movement)

#	Construction	+A	–A
55	AuxSVO	#43	#43
56		#44	#44
57		#45	#45
58		#46	#46
59	SAuxVO	Y ti sy'n coginio cinio heno.	Y ti'n coginio cinio heno.
60		Y ti sy'n rhoi'r ddarlith.	Y ti'n rhoi'r ddarlith.
61		Y ti sy'n casglu'r plant o'r ysgol.	Y ti'n casglu'r plant o'r ysgol.
62		Y ti sy'n dod â photel o win.	Y ti'n dod â photel o win.
63	VOAuxS	Ffonio'r gwasanaeth tân wyt ti.	Ffonio'r gwasanaeth tân ti.
64		Mynd i'r cigydd wyt ti.	Mynd i'r cigydd ti.
65		Siarad efo ffrind wyt ti.	Siarad efo ffrind ti.
66		Ymarfer Karate wyt ti.	Ymarfer Karate ti.
67	OAuxSV	Dillad wyt ti'n prynu.	Dillad ti'n prynu.

68	I'r canolfan hamdden wyt ti'n mynd.	I'r canolfan hamdden ti'n mynd.
69	Paned wyt ti'n yfed.	Paned ti'n yfed.
70	Yr heddlu wyt ti'n osgoi.	Yr heddlu ti'n osgoi.

GROUP V (Subordinates)

#	Construction	+A	–A
71	–infl	Dw i'n meddwl fod ti'n gyrru yn dda.	Dw i'n meddwl ti'n gyrru yn dda.
72		Mae'n bosib fod ti'n gwneud gormod.	Mae'n bosib ti'n gwneud gormod.
73		Mae Alun yn credu fod ti'n gofyn am lawer.	Mae Alun yn credu ti'n gofyn am lawer.
74		Dan ni'n gobeithio fod ti'n medru enill.	Dan ni'n gobeithio ti'n medru enill.

GROUP VI (Wh-Questions)

#	Construction	+A	–A
75	WH	Sut wyt ti'n mynd i Wrecsam?	Sut ti'n mynd i Wrecsam?
76		Be' wyt ti'n deud wrth Gwen am y ddamwain?	Be' ti'n deud wrth Gwen am y ddamwain?
77		Pwy wyt ti'n ei warhodd?	Pwy ti'n ei warhodd?
78		Pryd wyt ti'n symyd i'r Drenewydd?	Pryd ti'n symyd i'r Drenewydd?
79	WH+Prep	Lle wyt ti'n mynd i?	Lle ti'n mynd i?
80		Be' wyt ti'n golchi dy gi efo?	Lle ti'n golchi dy gi efo?
81		Lle wyt ti'n mynd i yn nes ymlaen?	Lle ti'n mynd i yn nes ymlaen?
82		Pwy wyt ti'n siarad efo?	Pwy ti'n siarad efo?
83	Prep+WH	O le wyt ti'n dod yn wreiddiol?	O le ti'n dod yn wreiddiol?
84		Efo pwy wyt ti'n dawnsio?	Efo pwy ti'n dawnsio?
85		Ers pryd wyt ti'n byw yma?	Ers pryd ti'n byw yma?

86	Ar pa' gwch wyt ti'n mynd?	Ar pa' gwch ti'n mynd?

C Program Code Listings for Judgement Experiment

Listing 3: *OpenSesame* script for judgement experiment

```
# Generated by OpenSesame 0.26 (Earnest Einstein)
# Thu Apr 26 19:59:57 2012 (nt)
#
# Copyright Sebastiaan Mathot (2010-2011)
# <http://www.cogsci.nl>
#
# NOTE: This script was edited in order to skip several hundred lines which are
#       automatically generated by the script waveDur.php. The two points at
#       which this happened (the training and experimental loops) have been
#       marked by a comment with reference to that script's output in this
#       source file and need to be replaced with the script's output before the
#       experiment can be run.

set foreground "white"
set subject_parity "even"
set description "Default description"
set title "AD Experiment"
set sampler_backend "legacy"
set coordinates "relative"
set height "768"
set mouse_backend "psycho"
set width "1024"
set compensation "0"
set keyboard_backend "psycho"
set background "black"
set subject_nr "0"
set canvas_backend "psycho"
set start "experiment"
set synth_backend "legacy"

define inline_script set_response_timeout
        set _run ""
        ___prepare__
        cue_duration = self.get("cue_duration")
        self.experiment.set("response_timeout", cue_duration+1500)
        __end__
        set description "Executes Python code"

define inline_script stop_playback
        ___run__
        import pygame
        pygame.mixer.stop()
        __end__
        set _prepare ""
        set description "Executes Python code"

define text_display instructions_1
        set foreground "white"
        set font_size "18"
```

```
50          set description "Presents a display consisting of text"
51          set maxchar "50"
52          set align "center"
53          __content__
54          Welcome to the auditory judgement task!
55
56          During this experiment, you will hear a short beep followed by a
                sentence in Welsh.
57          Some of these sentences are perfectly fine colloquial Welsh sentences,
                as you could possibly hear them somewhere in the street. However,
                some of the sentences were changed and probably don't sound right to
                 you.
58
59          Your task is to listen carefully to all the sentence and decide as
                quickly as you can whether you think that what you've just heard is
                an acceptable example of a colloquial Welsh sentence or not.
60          If you think it is okay, you should press the right (M) key - but if
                you think it doesn't really feel right to you, press the left (Z)
                key!
61
62          (Press any key for more instructions...)
63          __end__
64          set background "black"
65          set duration "keypress"
66          set font_family "mono"
67
68  define text_display instructions_2
69          set foreground "white"
70          set font_size "18"
71          set description "Presents a display consisting of text"
72          set maxchar "50"
73          set align "center"
74          __content__
75          Don't worry whether you think the sentences are "proper Welsh" - most
                of them aren't, and we don't really care. What we want to know about
                 is your personal intuition, what you would think if you heard this
                in real life. So remember that you are the real expert in this
                experiment!
76
77          We will now first give you 10 sentences to practice, as this task takes
                 a little getting used to at first. After this you will have the
                chance to take a little break (as at several points during the
                experiment!) before the real thing starts.
78          Should you have any problems you can ask the researcher for help during
                 the break.
79
80          To start the practice session press any key...
81          __end__
82          set background "black"
83          set duration "keypress"
84          set font_family "mono"
85
86  define sketchpad show_keys
87          set duration "0"
88          set description "Displays stimuli"
89          set start_response_interval "yes"
90          draw image -384 288 "cross.png" scale=1 center=1 show_if="always"
91          draw image 416 288 "check.png" scale=1 center=1 show_if="always"
92          draw textline -384 352 "Z" center=1 color=white font_family=mono
```

```
            font_size=18 show_if="always"
93          draw textline 416 352 "M" center=1 color=white font_family=mono
            font_size=18 show_if="always"
94
95  define inline_script experimental_loop_count
96          set _run ""
97          ____prepare__
98          # Loop counter
99          if self.has("experimental_loop_counter"):
100                 loop_counter = self.get("experimental_loop_counter")
101                 self.experiment.set("experimental_loop_counter", loop_counter
                        +1)
102         else:
103                 self.experiment.set("experimental_loop_counter", 0)
104         __end__
105         set description "Executes Python code"
106
107 define text_display thank_you
108         set foreground "white"
109         set font_size "18"
110         set description "Presents a display consisting of text"
111         set maxchar "50"
112         set align "center"
113         __content__
114         Thank you!
115
116         You've now completed all the items for this task.
117
118         Please let the researcher know that you're done.
119         __end__
120         set background "black"
121         set duration "keypress"
122         set font_family "mono"
123
124 define sequence stimulus_presentation
125         run set_response_timeout "always"
126         run fixation_dot "always"
127         run pre_beep_delay "always"
128         run beep "always"
129         run post_beep_delay "always"
130         run show_keys "always"
131         run stimuli "always"
132         run keyboard_response "always"
133         run stop_playback "always"
134         run logger "always"
135
136 define loop training_loop
137         set repeat "1"
138         set description "Repeatedly runs another item"
139         set skip "0"
140         set offset "no"
141         set item "stimulus_presentation"
142         set column_order "cue_no;cue_condition;cue_file;cue_duration"
143         set cycles "10"
144         set order "random"
145 #
146 # INSERT RESULTS FROM WAVEDUR.PHP SCRIPT FOR ./TRAIN DIRECTORY HERE
147 #
148         run stimulus_presentation
```

```
149
150 define sketchpad take_a_break_2
151         set duration "keypress"
152         set description "Displays stimuli"
153         set start_response_interval "no"
154         draw textline 0 -96 "Well done! You've completed [
                experimental_loop_counter]/152 items now." center=1 color=white
                font_family=mono font_size=18 show_if="always"
155         draw textline 0 -32 "Time to take a little break..." center=1 color=
                white font_family=mono font_size=18 show_if="always"
156         draw textline 0 32 "Just press any button to continue when you're ready
                !" center=1 color=white font_family=mono font_size=18 show_if="
                always"
157
158 define loop experimental_loop
159         set item "sequence"
160         set cycles "152"
161         set column_order "cue_no;cue_condition;cue_file;cue_duration"
162 #
163 # INSERT RESULTS FROM WAVEDUR.PHP SCRIPT FOR ./FINAL DIRECTORY HERE
164 #
165         run experimental_sequence
166
167 define sampler beep
168         set volume "0.3"
169         set description "Plays a sound file in .wav or .ogg format"
170         set sample "beep.wav"
171         set pitch "1"
172         set duration "sound"
173         set stop_after "0"
174         set pan "0"
175         set fade_in "0"
176
177 define sequence experiment
178         run training "always"
179         run experimental_loop "always"
180         run thank_you "always"
181         run exit "always"
182
183 define keyboard_response exit
184         set allowed_responses "q"
185         set description "Collects keyboard responses"
186         set timeout "infinite"
187         set flush "yes"
188
189 define advanced_delay post_beep_delay
190         set duration "100"
191         set jitter "0"
192         set description "Waits for a specified duration"
193         set jitter_mode "Std. Dev."
194
195 define fixation_dot fixation_dot
196         set foreground "white"
197         set style "cross"
198         set description "Presents a central fixation dot with a choice of
                various styles"
199         set y "0"
200         set background "black"
201         set duration "500"
```

```
202         set x "0"
203         set penwidth "3"
204
205 define logger logger
206         set ignore_missing "yes"
207         set unicode "no"
208         set description "Logs experimental data"
209         set auto_log "no"
210         set use_quotes "yes"
211         log "cue_duration"
212         log "cue_condition"
213         log "cue_file"
214         log "cue_no"
215         log "response_keyboard_response"
216         log "response_time_keyboard_response"
217
218 define sequence experimental_sequence
219         run experimental_loop_count "always"
220         run take_a_break "[experimental_loop_counter] = 38"
221         run take_a_break "[experimental_loop_counter] = 76"
222         run take_a_break "[experimental_loop_counter] = 114"
223         run stimulus_presentation "always"
224
225 define advanced_delay pre_beep_delay
226         set duration "100"
227         set jitter "0"
228         set description "Waits for a specified duration"
229         set jitter_mode "Std. Dev."
230
231 define text_display end_of_training
232         set foreground "white"
233         set font_size "18"
234         set description "Presents a display consisting of text"
235         set maxchar "50"
236         set align "center"
237         __content__
238         Well done, you've completed the training task.
239         Feel free to take a little break now!
240
241         When you think you are ready just press any key to start the experiment
              .
242         __end__
243         set background "black"
244         set duration "keypress"
245         set font_family "mono"
246
247 define sequence training
248         run instructions_1 "always"
249         run instructions_2 "always"
250         run training_loop "always"
251         run end_of_training "always"
252
253 define text_display take_a_break
254         set foreground "white"
255         set font_size "18"
256         set description "Presents a display consisting of text"
257         set maxchar "50"
258         set align "center"
259         __content__
```

```
260         Well done! You've completed the first [experimental_loop_counter] out
                of 152 items.
261
262         Time to take a little break...
263
264         Press any button to continue with the experiment when you are ready!
265         __end__
266         set background "black"
267         set flush "yes"
268         set duration "keypress"
269         set font_family "mono"
270
271 define sampler stimuli
272         set sample "./audio/[cue_file].wav"
273         set description "Plays a sound file in .wav or .ogg format"
274         set volume "1"
275         set timeout "0"
276         set pitch "1"
277         set duration "0"
278         set stop_after "0"
279         set pan "0"
280         set fade_in "0"
281
282 define keyboard_response keyboard_response
283         set allowed_responses "z;m"
284         set description "Collects keyboard responses"
285         set timeout "[response_timeout]"
286         set flush "yes"
```

Listing 4: Script for calculating duration of waveform files

```
1  <?php
2  /***
3   * Generate List of Audio Stimuli and Durations for OpenSesame
4   *
5   * This script parses either the directory ./TRAIN or ./FINAL for waveform audio
6   * files (.wav) and then generates a list with their filenames and their
7   * duration in milliseconds which can be pasted into OpenSesame's loop tables.
8   * Takes one command line argument, either TRAIN or FINAL to determine the set
9   * of files to be processed.
10  *
11  * PHP Version 5.3
12  *
13  * LICENSE: This piece of software was developed as part of a BA (Hons)
14  * dissertation at Bangor University. It may be freely distributed and used by
15  * anybody whomsoever, so long as the author is acknowledged and no changes
16  * are made to the source code without prior agreement with the author.
17  *
18  * @author      Florian Breit <f.breit@univ.bangor.ac.uk>
19  * @copyright   2012 Florian Breit
20  * @version     1.0.0
21  */
22
23 //
24 // SETUP
25 //
26
27 //Some PHP stuff
```

```
28 error_reporting(E_ALL);
29 ini_set('display_errors', 1);
30
31 //Paths
32 $root_path = "./";
33
34 //
35 // MAIN SCRIPT
36 //
37
38 //Check command line argument is okay
39 if(isset($argv[1]) && ($argv[1] == "TRAIN" || $argv[1] == "FINAL") ) {
40     $type = $argv[1];
41 } else {
42     die("First argument must be either TRAIN or FINAL.");
43 }
44
45 //Scan all files and print info
46 $dir = scandir($root_path.$type);
47 $i = 0;
48 foreach($dir as $file) {
49     if( substr($file, -4) == ".wav" ) {
50         $dur = (int) wavDur("./$type/".$file);
51         $cond = substr($file, 2, 2);
52         $no = substr($file, 0, 2);
53         print "setcycle $i cue_no \"$no\"\n";
54         print "setcycle $i cue_condition \"$cond\"\n";
55         print "setcycle $i cue_file \"$type/$no$cond\"\n";
56         print "setcycle $i cue_duration \"$dur\"\n";
57         $i++;
58     }
59 }
60
61 //
62 // FUNCTIONS
63 //
64
65 /***
66  * Read Header and Duration from RIFF Waveform Files
67  *
68  * This function reads the header information from a RIFF Waveform file and
69  * then calculates the file duration (for the audio content) in milliseconds.
70  * The function is adapted from an earlier code snippet posted by "valinea"
71  * on 07 August 2006 at http://snipplr.com/view/285/.
72  *
73  * @link http://snipplr.com/view/285/
74  * @author valinea (http://snipplr.com/users/velinea/)
75  * @param string $file Path to the waveform file to be analysed
76  * @return int Returns the duration of the waveform in seconds.
77  */
78 function wavDur($file) {
79     $fp = fopen($file, 'r');
80     if(fread($fp,4) == "RIFF") {
81         fseek($fp, 20);
82         $rawheader = fread($fp, 16);
83         $packing = 'vtype/vchannels/Vsamplerate/Vbytespersec/valignment/vbits';
84         $header = unpack($packing, $rawheader);
85         $pos = ftell($fp);
86         while(fread($fp,4) != "data" && !feof($fp)) {
```

```
87              $pos++;
88              fseek($fp,$pos);
89          }
90          $rawheader = fread($fp, 4);
91          $data = unpack('Vdatasize',$rawheader);
92          $sec = $data[datasize]/$header[bytespersec];
93          $ms = $sec*1000;
94          return $ms;
95      }
96  }
97  ?>
```

Listing 5: Script for generating offline questionnaire

```
1  <?php
2  /***
3   * Generate Randomised Questionnaire
4   *
5   * This script reads a list of stimuli and then creates a HTML file containing
6   * a set of instructions and the list of stimuli in pseudo randomised order with
7   * a Likert scale from 1 to 5 and checkboxes for each item, split into blocks of
8   * 30, with the first block drawn from a set of training stimuli. CConditions
9   * are masked by a special function containing partially random digits so that
10  * participants are unlikely to discover a pattern in stimuli numeration. See
11  * the file stimuli.php for an example of the kind of stimuli list required.
12  *
13  * PHP Version 5.3
14  *
15  * LICENSE: This piece of software was developed as part of a BA (Hons)
16  * dissertation at Bangor University. It may be freely distributed and used by
17  * anybody whomsoever, so long as the author is acknowledged and no changes
18  * are made to the source code without prior agreement with the author.
19  *
20  * @author      Florian Breit <f.breit@univ.bangor.ac.uk>
21  * @copyright   2012 Florian Breit
22  * @version     1.0.0
23  */
24
25
26  //
27  // SETUP
28  //
29
30  //Some PHP stuff
31  error_reporting(E_ALL);
32  ini_set('display_errors', 1);
33
34  //Paths
35  $stimuli_list = './stimuli.php';
36  $output_file = './quest.html';
37
38  //
39  // MAIN SCRIPT
40  //
41
42  //Fetch stimuli list
43  require($stimuli_list);
44  if(!isset($TRAIN) || !isset($FINAL)) {
```

```
45      die('Stimuli set is either missing $TRAIN or $FINAL data structures.');
46  }
47
48  //Start output buffering and print HTML header
49  ob_start();
50  print '<?xml version="1.0" encoding="UTF-8"?>';
51  print <<<HTML
52  <html>
53      <head>
54          <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
55          <title>AD EXPERIMENT &mdash; OFFLINE JUDGEMENT TASK</title>
56          <style>
57              body {
58                  font-size:12pt;
59                  font-family:serif;
60                  padding:0px;
61                  margin:0px;
62              }
63              h1 {
64                  font-size:150%;
65              }
66              h2 {
67                  font-size:115%;
68                  page-break-before:auto;
69                  page-break-after:avoid;
70              }
71              h2.break {
72                  page-break-before:always;
73              }
74              div p {
75                  display:inline-block;
76                  margin:2px;
77              }
78              div p.id {
79                  width:auto;
80                  font-size:60%;
81                  font-family:monospace;
82                  color:#777;
83              }
84              div p.stimuli {
85                  width:70%;
86              }
87              div p.likert {
88                  width:auto;
89              }
90              div p.likert span {
91                  display:inline-block;
92                  width:22px;
93                  text-align:center;
94              }
95          </style>
96      </head>
97      <body>
98          <h1>AD EXPERIMENT &mdash; OFFLINE JUDGEMENT TASK</h1>
99          <h2>Instructions</h2>
100         <p>
101             In this task you are presented with a number of colloquial Welsh
102             sentences similar to those you've heared in the computer-aided task
103             you have just completed. Again some of these sentences will be just
```

```
104             fine and some of them will probably seem rather odd to you.
105         </p>
106         <p>
107             As opposed to the previous task however, this time we want you to
108             rate how acceptable the given sentences seem to you. For this you
109             will see a five point scale next to every sentence. You should use
110             the number 1 to indicate that you feel the sentence is completely
111             unacceptable and the number 5 to indicate that it feels completely
112             acceptable to you. Use any of the numbers in-between to indicate
113             that you have a tendency to say it is acceptable or unacceptable, or
114             the box in the middle if you cannot decide at all.
115         </p>
116         <p>
117             Again this is about what you feel is appropriate in the colloquial,
118             spoken language and that this is not about what you may have been
119             taught about Welsh in school. As you will surely know sometimes what
120             people do can be very different from what they teach! So remember
121             that this is about your personal opinion about the language you
122             speak and so you are the real expert!
123         </p>
124
125 HTML;
126
127 //Print likert scale for training data
128 print <<<HTML
129         <h2>Block 1</h2>
130         <div>
131             <p class="id">    </p>
132             <p class="stimuli"></p>
133             <p class="likert">
134                 <span>1</span>
135                 <span>2</span>
136                 <span>3</span>
137                 <span>4</span>
138                 <span>5</span>
139             </p>
140         </div>
141
142 HTML;
143
144 //Randomise and print training stimuli
145 $training_data = rand_stimuli($TRAIN);
146 foreach($training_data as $stimulus) {
147     $id = hide_id($stimulus[0], $stimulus[1]);
148     $sentence = $stimulus[2];
149     print <<<HTML
150         <div>
151             <p class="id">$id</p>
152             <p class="stimuli">$sentence</p>
153             <p class="likert">
154                 <span>&#9744;</span>
155                 <span>&#9744;</span>
156                 <span>&#9744;</span>
157                 <span>&#9744;</span>
158                 <span>&#9744;</span>
159             </p>
160         </div>
161
162 HTML;
```

```
163 }
164
165 //Randomise experimental stimuli
166 $experimental_data = rand_stimuli($FINAL);
167 $block_counter = 2; //Block 1 were the Training stimuli
168
169 //Print experimental stimuli in blocks of 30
170 for($i=0; $i< count($experimental_data); $i++) {
171     if($i%30 == 0) {
172         //Print block header with numbers for likert scale
173         print <<<HTML
174         <h2 class="break">Block $block_counter</h2>
175         <div>
176             <p class="id">    </p>
177             <p class="stimuli"></p>
178             <p class="likert">
179                 <span>1</span>
180                 <span>2</span>
181                 <span>3</span>
182                 <span>4</span>
183                 <span>5</span>
184             </p>
185         </div>
186
187 HTML;
188         $block_counter++;
189     }
190     $stimulus = $experimental_data[$i];
191     $id = hide_id($stimulus[0], $stimulus[1]);
192     $sentence = $stimulus[2];
193     //Print stimuli and likert scale
194     print <<<HTML
195         <div>
196             <p class="id">$id</p>
197             <p class="stimuli">$sentence</p>
198             <p class="likert">
199                 <span>&#9744;</span>
200                 <span>&#9744;</span>
201                 <span>&#9744;</span>
202                 <span>&#9744;</span>
203                 <span>&#9744;</span>
204             </p>
205         </div>
206
207 HTML;
208 }
209
210 //Print HTML footer
211 print <<<HTML
212     </body>
213 </html>
214 HTML;
215
216 //Write output buffer to output file
217 $ob = ob_get_contents();
218 $fh =fopen($output_file, 'w+');
219 fwrite($fh, $ob);
220 fclose($fh);
221
```

```
222 //NB: Output buffer wil be flushed to STDOUT at end of script!
223
224 //
225 // FUNCTIONS
226 //
227
228 /***
229  * Randomise list of stimuli
230  *
231  * This function takes an array of stimuli in two conditions and merges these
232  * into one single list, assigning a pseudo random order to every single item.
233  *
234  * @param array $stimuli A two-dimensional array of stimuli to be randomised
235  * @return array Returns a flat array of stimuli in pseudo random order
236  */
237 function rand_stimuli($stimuli) {
238     $keys = array_keys($stimuli);        //get all keys
239     $keys = array_merge($keys, $keys);   //double keys (+A and -A conditions)
240     shuffle($keys);                      //pseudo-randomisation
241     $result = array();
242     foreach($keys as $key) {
243         if(count($stimuli[$key]) > 1) {
244             $cond = rand(0, 1) ? 'ad' : 'oa'; //pseudo-random +A or -A
245             if(substr($stimuli[$key][$cond], 0, 1) == '#') {
246                 continue;   //skip references to #xx
247             }
248             $result[] = array($key, $cond, $stimuli[$key][$cond]);
249             unset($stimuli[$key][$cond]);
250         } else {
251             $cond = array_keys($stimuli[$key]); //either oa or ad
252             $cond = $cond[0];
253             if(substr($stimuli[$key][$cond], 0, 1) == '#') {
254                 continue;   //skip references to #xx
255             }
256             $result[] = array($key, $cond, $stimuli[$key][$cond]);
257             unset($stimuli[$key][$cond]);
258         }
259     }
260
261     return $result;
262 }
263
264 /***
265  * Hide Stimuli ID and condition
266  *
267  * This function takes the id and condition description (oa or ad) from a
268  * stimuli and returns a string masking these in some predetermined pseudo
269  * random numbers, which can be converted back into the original stimuli
270  * ID and condition. Specifically a pseudo random number between 0 and 4 is
271  * assigned to the condition 'ad' and one between 5 and 9 to 'oa'. This is
272  * followed by the stimuli ID, with a leading zero where applicable. Another
273  * pseudo random number between 0 and 9 is appended at the end.
274  *
275  * @param string $cond A string indicating the experimental condition, either
276  *                     'oa' (+A) or 'ad' (-A)
277  * @param int $id The stimuli ID
278  * @return string Returns a string of numbers encoding condition and ID
279  */
280 function hide_id($id, $cond) {
```

```
281     if($cond == 'ad') {
282         $p = rand(0, 4); //0-4 to mark -A
283     } else {
284         $p = rand(5, 9); //5-9 to mark +A
285     }
286     if(strlen($id) == 1) {
287         $id = '0'.$id;  //enforce preceeding 0
288     }
289     $t = rand(0, 9);     //random trailing number
290
291     return $p.$id.$t;
292 }
293 ?>
```

Listing 6: Sample format of stimuli.php file

```
 1 <?php
 2 /***
 3  * Sample of Format for stimuli.php
 4  *
 5  * This file contains a sample of the formatting in which the file
 6  * stimuli.php, used by the make_questionnaire.php script, should be.
 7  * This should contain two arrays, $TRAIN for the training stimuli and
 8  * $FINAL for the test stimuli, both following the format indicated
 9  * below. The text may include references to other stimuli in the form
10  * #NN giving their index number in the array, these will be skipped in
11  * the final output.
12  *
13  * PHP Version 5.3
14  *
15  * LICENSE: This piece of software was developed as part of a BA (Hons)
16  * dissertation at Bangor University. It may be freely distributed and used by
17  * anybody whomsoever, so long as the author is acknowledged and no changes
18  * are made to the source code without prior agreement with the author.
19  *
20  * @author      Florian Breit <f.breit@univ.bangor.ac.uk>
21  * @copyright   2012 Florian Breit
22  * @version     1.0.0
23  */
24
25 $TRAIN = array(
26       1 => array(
27               'oa' => "Dwyt Wyn ddim isio yfed bara brith o gwbl!",
28               'ad' => "Wyn ddim isio yfed bara brith o gwbl!",
29               ),
30       2 => array(
31               'oa' => "Next sentence in overt auxiliary condition",
32               'ad' => "Next sentence in auxiliary deletion condition",
33               ),
34       3 => array(
35               'oa' => "...",
36               'ad' => "...,"
37               ),
38       //etc..
39 );
40
41 ?>
```

Listing 7: Script for entering background data

```
<html>
    <head>
        <title>Participant Background Data</title>
        <style type="text/css">
            input {
                margin:1px;
                padding:2px;
                border:1px solid #999;
            }
            input:active, input:focus {
                border:1px solid black;
                background:#FFA;
            }
            label[for] {
                font-weight: bold;
                min-width:40px;
                display:inline-block;
            }
        </style>
    </head>
    <body>
        <?php
        error_reporting(E_ALL);
        if(!empty($_POST)) {
            $code = str_replace(array('\\', '/'), '', $_POST['code']);
            $fh = fopen("./results/background-$code.csv", 'w+');
            fwrite($fh, "age,gender,education,wherefrom,southnorth\r\n");
            $age = $_POST['age'];
            $gender = $_POST['gender'];
            $education = $_POST['education'];
            $wherefrom = $_POST['wherefrom'];
            $southnorth = $_POST['southnorth'];
            fwrite($fh, "\"$age\",");
            fwrite($fh, "\"$gender\",");
            fwrite($fh, "\"$education\",");
            fwrite($fh, "\"$wherefrom\",");
            fwrite($fh, "\"$southnorth\"");
            fclose($fh);
            print("Data written to file: background-$code.csv<br />");
            print("<iframe src='./results/background-$code.csv'></iframe><br />");
        }
        ?>
        <h1>Participant Background Data</h1>
        <form method="post">
            <label for="age">Code</label>
                <input type="text" name="code" size="4" /><br />
            <label for="age">Age</label>
                <input type="text" name="age" size="2" /><br />
            <label for="gender">Gender</label><br />
                <input type="radio" name="gender" value="1" />
                    <label>Male</label><br />
                <input type="radio" name="gender" value="2" />
                    <label>Female</label><br />
            <label for="education">Education</label><br />
                <input type="radio" name="education" value="1" />
                    <label>GCSEs</label><br />
                <input type="radio" name="education" value="2" />
```

```
58                      <label>AS/A-Levels</label><br />
59                  <input type="radio" name="education" value="3" />
60                      <label>(Some) HE</label><br />
61                  <input type="radio" name="education" value="4" />
62                      <label>(Some) PG Ed</label><br />
63              <label for="wherefrom">Where are you from?</label>
64                  <input type="text" name="wherefrom" size="20" /><br />
65              <label for="southnorth">South/Mid/North-Walian?</label><br />
66                  <input type="radio" name="southnorth" value="1" />
67                      <label>South</label><br />
68                  <input type="radio" name="southnorth" value="2" />
69                      <label>Mid</label><br />
70                  <input type="radio" name="southnorth" value="3" />
71                      <label>North</label><br />
72              <button type="submit">Submit</button>
73          </form>
74      </body>
75  </html>
```

Listing 8: Script for entering questionnaire results

```
1  <?php
2
3  function resolve_id($hidden_id) {
4      //resolve condition (ad < 5 >= od)
5      $cond = substr($hidden_id, 0, 1);
6      if($cond < 5) {
7          $cond = 'ad';
8      } else {
9          $cond = 'oa';
10     }
11     //extract id
12     $id = (int) substr($hidden_id, 1, 2);
13
14     return array('id' => $id,
15                  'cond' => $cond,
16                   0 => $id,
17                   1 => $cond);
18 }
19
20 ?>
21 <html>
22     <head>
23         <title>Offline Task Data Sheet</title>
24         <style type="text/css">
25             .id {
26                 width:40px;
27             }
28             .value {
29                 width:15px;
30             }
31             input {
32                 margin:1px;
33                 padding:2px;
34             }
35             input:active, input:focus {
36                 border:1px solid black;
37                 background:#FFA;
```

```
38              }
39          </style>
40      </head>
41      <body>
42          <?php
43          error_reporting(E_ALL);
44          $train_results = array();
45          $final_results = array();
46          if(!empty($_POST)) {
47              $code = str_replace(array('\\', '/'), '', $_POST['code']);
48              $fh = fopen("./results/offline-train-$code.csv", 'w+');
49              fwrite($fh, "id,cond,rating\r\n");
50              foreach($_POST['train_id'] as $index => $id) {
51                  list($id, $cond) = resolve_id($id);
52                  $value = $_POST['train_value'][$index];
53                  $train_results[$id][$cond] = $value;
54                  fwrite($fh, "\"$id\",\"$cond\",\"$value\"\r\n");
55              }
56              fclose($fh);
57              print("Data written to file: offline-train-$code.csv<br />");
58              print("<iframe src='./results/offline-train-$code.csv'></iframe><br />");
59              $fh = fopen("./results/offline-$code.csv", 'w+');
60              fwrite($fh, "id,cond,rating\r\n");
61              foreach($_POST['id'] as $index => $id) {
62                  list($id, $cond) = resolve_id($id);
63                  $value = $_POST['value'][$index];
64                  $final_results[$id][$cond] = $value;
65                  fwrite($fh, "\"$id\",\"$cond\",\"$value\"\r\n");
66              }
67              fclose($fh);
68              print("Data written to file: offline-$code.csv<br />");
69              print("<iframe src='./results/offline-$code.csv'></iframe><br />");
70          }
71          ?>
72          <form method="post">
73              <p>
74                  <b>Code:</b> <input type="text" name="code" />
75              </p>
76              <h2>Block 1</h2>
77                  <ol>
78                      <?php for($i=0;$i<10;$i++) { ?>
79                      <li>
80                          <input type="text" class="id" name="train_id[]" />
81                          <input type="text" class="value" name="train_value[]" />
82                      </li>
83                      <?php } ?>
84                  </ol>
85              <?php for($block=2;$block<=6;$block++) { ?>
86                  <h2>Block <?=$block?></h2>
87                      <ol>
88                          <?php for($i=0;$i<30;$i++) { ?>
89                          <li>
90                              <input type="text" class="id" name="id[]" />
91                              <input type="text" class="value" name="value[]" />
92                          </li>
93                          <?php } ?>
94                      </ol>
95              <?php } ?>
96              <button type-"submit">Submit</button>
```

```
97          </form>
98      </body>
99  </html>
```

Listing 9: Script for merging results from online and offline tasks

```
1  <?php
2  /***
3   * Merge Results from Online and Offline Tasks
4   *
5   * This script merges the CSV files generated by the OpenSesame experiment for
6   * the online task with the data typed up from the offline task and the survey
7   * on participant's background data via the background_data.php and
8   * offline_results.php scripts. Merged results are stored in an SQLite database.
9   *
10  * PHP Version 5.3
11  *
12  * LICENSE: This piece of software was developed as part of a BA (Hons)
13  * dissertation at Bangor University. It may be freely distributed and used by
14  * anybody whomsoever, so long as the author is acknowledged and no changes
15  * are made to the source code without prior agreement with the author.
16  *
17  * @author      Florian Breit <f.breit@univ.bangor.ac.uk>
18  * @copyright   2012 Florian Breit
19  * @version     1.0.0
20  */
21
22
23 //
24 // SETUP
25 //
26
27 //Some PHP stuff
28 error_reporting(E_ALL);
29 ini_set('display_errors', 1);
30
31 //Paths
32 $results_path = './';
33 $db_path = './judgement_data.sqlite';
34
35 //
36 // MAIN SCRIPT
37 //
38
39 //Set up and flush database
40 $fh = @fopen($db_path, 'w'); //Flushes DB
41 if( $fh === false ) {
42     die("\nError: Could not open file "
43         ."'$db_path' for writing.");
44 }
45 fclose($fh);
46 $db = new SQLite3($db_path, SQLITE3_OPEN_READWRITE);
47 $result = $db->exec("CREATE TABLE participants
48                              (
49                                  p_id            INTEGER PRIMARY KEY,
50                                  p_code          TEXT,
51                                  p_age           INTEGER,
52                                  p_gender        INTEGER,
```

```
53                              p_education     INTEGER,
54                              p_wherefrom     TEXT,
55                              p_southnorth    INTEGER
56                          );
57                      CREATE TABLE results
58                          (
59                              p_id            INTEGER,
60                              s_id            INTEGER,
61                              s_cond          INTEGER,
62                              s_duration      INTEGER,
63                              r_on_response   INTEGER,
64                              r_on_rtime      INTEGER,
65                              r_off_rating    INTEGER
66                          );
67                      ");
68
69 //Scan results directory (directory with the CSV files)
70 $dir = scandir($results_path);
71 //Walk through files and insert their contents into db
72 foreach($dir as $file) {
73     //Only files starting with "subject" such as "subject-abc1.csv"
74     if( substr($file, 0, 7) == 'subject' ) {
75         //Find code for relevant files (subject-xxxx.csv -> xxxx)
76         $code = substr($file, 8, 4);
77         print "$code\n";
78
79         //Read data from all files with $code
80         $online_data = read_csv($results_path."/subject-$code.csv");
81         $offline_data = read_csv($results_path."/offline-$code.csv");
82         //$offline_train = read_csv($results_path."/offline-train-$code.csv");
83         $background_data =read_csv($results_path."/background-$code.csv");
84         //Add data to database
85
86         //Add participant background data
87         $stmt = $db->prepare('INSERT INTO participants
88                                 (
89                                     p_code,
90                                     p_age,
91                                     p_gender,
92                                     p_education,
93                                     p_wherefrom,
94                                     p_southnorth
95                                 )
96                                 VALUES
97                                 (
98                                     :p_code,
99                                     :p_age,
100                                    :p_gender,
101                                    :p_education,
102                                    :p_wherefrom,
103                                    :p_southnorth
104                                )
105                            ');
106         $age = $background_data[1][0];
107         $gender = $background_data[1][1];
108         $education = $background_data[1][2];
109         $wherefrom = $background_data[1][3];
110         $southnorth = $background_data[1][4];
111         $stmt->reset();
```

```
112         $stmt->bindValue(':p_code',          $code);
113         $stmt->bindValue(':p_age',           $age);
114         $stmt->bindValue(':p_gender',        $gender);
115         $stmt->bindValue(':p_education',     $education);
116         $stmt->bindValue(':p_wherefrom',     $wherefrom);
117         $stmt->bindValue(':p_southnorth',    $southnorth);
118         $stmt->execute();
119         $p_id = $db->lastInsertRowID();
120
121         //Add online results
122         $stmt = $db->prepare('INSERT INTO results
123                               (
124                                   p_id,
125                                   s_id,
126                                   s_cond,
127                                   s_duration,
128                                   r_on_response,
129                                   r_on_rtime
130                               )
131                               VALUES
132                               (
133                                   :p_id,
134                                   :s_id,
135                                   :s_cond,
136                                   :s_duration,
137                                   :r_on_response,
138                                   :r_on_rtime
139                               );
140                             ');
141         for($i=1;$i<count($online_data);$i++) {
142             //ignore training data
143             if( substr($online_data[$i][2], 0, 5) == 'TRAIN' ) {
144                 continue;
145             }
146             if($online_data[$i][0] == 'oa') {
147                 $cond = 1;
148             } else {
149                 $cond = 2;
150             }
151             $dur = $online_data[$i][1];
152             $id = $online_data[$i][3];
153             if($online_data[$i][4] == 'z' ) {
154                 $resp = 1;
155             } elseif($online_data[$i][4] == 'm' ) {
156                 $resp = 2;
157             } else {
158                 $resp = null;
159             }
160             $rt = $online_data[$i][5];
161             if($rt == 'timeout') {
162                 $rt = null;
163             }
164             $stmt->reset();
165             $stmt->bindValue(':p_id',            $p_id);
166             $stmt->bindValue(':s_id',            $id);
167             $stmt->bindValue(':s_cond',          $cond);
168             $stmt->bindValue(':s_duration',      $dur);
169             $stmt->bindValue(':r_on_response',   $resp);
170             $stmt->bindValue(':r_on_rtime',      $rt);
```

```
171             $stmt->execute();
172         }
173
174         //Add offline results
175         $stmt = $db->prepare('UPDATE results
176                                 SET
177                                     r_off_rating = :r_off_rating
178                                 WHERE
179                                     p_id = :p_id
180                                     AND
181                                     s_id = :s_id
182                                     AND
183                                     s_cond = :s_cond
184                               ');
185         for($i=1;$i<count($offline_data);$i++) {
186             $s_id = $offline_data[$i][0];
187             if($s_id === '0') {
188                 $s_id = '9'; //correct for programming error
189             }
190             $s_cond = $offline_data[$i][1];
191             if($s_cond == 'oa') {
192                 $s_cond = 1;
193             } else {
194                 $s_cond = 2;
195             }
196             $r_off_rating = $offline_data[$i][2];
197             if(!is_numeric($r_off_rating)) {
198                 $r_off_rating = null;
199             }
200             $stmt->reset();
201             $stmt->bindValue(':p_id',           $p_id);
202             $stmt->bindValue(':r_off_rating',   $r_off_rating);
203             $stmt->bindValue(':s_id',           $s_id);
204             $stmt->bindValue(':s_cond',         $s_cond);
205             $stmt->execute();
206         }
207     }
208 }
209
210 //Create data views in the database
211 $result = $db->exec('CREATE VIEW
212                         combined
213                     AS
214                         SELECT
215                             participants.p_id,
216                             p_age,
217                             p_gender,
218                             p_education,
219                             p_southnorth,
220                             s_id,
221                             s_cond,
222                             s_duration,
223                             r_on_response,
224                             r_on_rtime,
225                             r_off_rating
226                         FROM
227                             participants,
228                             results
229                         WHERE
```

```
230                             participants.p_id = results.p_id
231                         ;');
232 $result = $db->exec('CREATE VIEW
233                             combined_per_sentence
234                         AS
235                             SELECT
236                                 s_id,
237                                 s_cond,
238                                 round(avg(r_on_response), 2)
239                                     AS avg_on_response,
240                                 round(avg(r_on_rtime), 2)
241                                     AS avg_on_rtime,
242                                 round(avg(r_on_rtime-s_duration), 2)
243                                     AS avg_on_score,
244                                 round(avg(r_off_rating), 2)
245                                     AS avg_off_rating
246                             FROM
247                                 results
248                             GROUP BY
249                                 s_id,
250                                 s_cond
251                         ;');
252
253 //
254 // FUNCTIONS
255 //
256
257 /***
258  * Read CSV file into array
259  *
260  * This function reads the specified CSV file, using the optionally defined
261  * separator (default ',') and using quotations to assign fields (default '"').
262  * The function returns a two-dimensional array containing the rows and fields
263  * present in the CSV file. An empty array is returned if the CSV file is empty.
264  *
265  * @param string $file Path to the CSV file to be read
266  * @param string $sep Separator for fields, default ','
267  * @param string $trim Characters to be trimmed from either side of fields
268  * @return array Returns a two-dimensional array representing rows and columns
269  */
270 function read_csv($file, $sep=',', $trim='"') {
271     $lines = file($file);
272     foreach($lines as $key => $line) {
273         $line = trim($line);
274         $quot = false;
275         $line = csv_explode($sep, $line);
276         foreach($line as $index => $value) {
277             $line[$index] = trim($value, $trim);
278         }
279         $lines[$key] = $line;
280     }
281     return $lines;
282 }
283
284 /***
285  * Explode CSV line into Array
286  *
287  * This function takes a line from a typical CSV file and separates it into an
288  * array using the given separator, much like explode(). However it ignores any
```

```
289  * occurences of the separator inside double quotation marks ('"').
290  *
291  * @param string $sep The separator to be used
292  * @param string $line The CSV line to be parsed
293  * @return array Returns an array with the individual fields in the CSV line
294  */
295 function csv_explode($sep, $line) {
296     $return = array();
297     $cell_count = 0;
298     $return[0] = '';
299     $quot = false;
300     for($i=0;$i<strlen($line);$i++) {
301         if($quot) {
302             if($line[$i] == '"') {
303                 $quot = false;
304             } else {
305                 //ignore sep until unquoting
306                 $return[$cell_count] .= $line[$i];
307             }
308         } else {
309             if($line[$i] == '"') {
310                 $quot = true;
311             } else {
312                 if($line[$i] == $sep) {
313                     $cell_count++;
314                     $return[$cell_count] = '';
315                 } else {
316                     $return[$cell_count] .= $line[$i];
317                 }
318             }
319         }
320     }
321     return $return;
322 }
323 ?>
```

D Instructions for Judgement Experiment

Instructions for Online Task

Welcome to the auditory judgement task!

During this experiment, you will hear a short beep followed by a sentence in Welsh. Some of these sentences are perfectly fine colloquial Welsh sentences, as you could possibly hear them somewhere in the street. However, some of the sentences were changed and probably don't sound right to you.

Your task is to listen carefully to all the sentence and decide as quickly as you can whether you think that what you've just heard is an acceptable example of a colloquial Welsh sentence or not. If you think it is okay, you should press the right (M) key - but if you think it doesn't really feel right to you, press the left (Z) key!

(Page Break)

Don't worry whether you think the sentences are "proper Welsh" - most of them aren't, and we don't really care. What we want to know about is your personal intuition, what you would think if you heard this in real life. So remember that you are the real expert in this experiment!

We will now first give you 10 sentences to practice, as this task takes a little getting used to at first. After this you will have the chance to take a little break (as at several points during the experiment!) before the real thing starts. Should you have any problems you can ask the researcher for help during the break.

To start the practice session press any key...

Instructions for Offline Task

In this task you are presented with a number of colloquial Welsh sentences similar to those you've heard in the computer-aided task you have just completed. Again some of these sentences will be just fine and some of them will probably seem rather odd to you.

As opposed to the previous task however, this time we want you to rate how acceptable the given sentences seem to you. For this you will see a five point scale next to every sentence. You should use the number 1 to indicate that you feel the sentence is completely unacceptable and the number 5 to indicate that it feels completely acceptable to you. Use any of the numbers in-between to indicate that you have a tendency to say it is acceptable or unacceptable, or the box in the middle if you cannot decide at all.

Again this is about what you feel is appropriate in the colloquial, spoken language and that this is not about what you may have been taught about Welsh in school. As you will surely know sometimes what people do can be very different from what they teach! So remember that this is about your personal opinion about the language you speak and so you are the real expert!

E Sample of Offline Judgement Questionnaire

Block 2

		1	2	3	4	5
4610	Y ti'n casglu'r plant o'r ysgol.	☐	☐	☐	☐	☐
6773	Pwy wyt ti'n ei warhodd?	☐	☐	☐	☐	☐
4544	Ti ddim yn cael mynd adra eto.	☐	☐	☐	☐	☐
8509	Wyt ti'n dilyn Pobol y Cwm ar S4C?	☐	☐	☐	☐	☐
3670	Dillad ti'n prynu.	☐	☐	☐	☐	☐
2310	Ti neud dy waith cartref di yn dda ddoe.	☐	☐	☐	☐	☐
4497	Ti'n cael cawod heno?	☐	☐	☐	☐	☐
8662	Ymarfer Karate wyt ti.	☐	☐	☐	☐	☐
4402	Ti 'di golchi'r llestri.	☐	☐	☐	☐	☐
6653	Siarad efo ffrind wyt ti.	☐	☐	☐	☐	☐
8708	Yr heddlu wyt ti'n osgoi.	☐	☐	☐	☐	☐
7328	Wnest ti fwydo'r planhigion mwy nag wythnos yn ôl.	☐	☐	☐	☐	☐
3015	Fi'n licio hufen iâ.	☐	☐	☐	☐	☐
1355	Ti 'sgubo 'fory.	☐	☐	☐	☐	☐
0437	Ti'n chwarae tennis yn dda.	☐	☐	☐	☐	☐
4213	Ti'n ffonio fi neithiwr.	☐	☐	☐	☐	☐
2002	Ni'n neidio o gwmpas ar y gwely.	☐	☐	☐	☐	☐
0500	Ti'n dilyn Pobol y Cwm ar S4C?	☐	☐	☐	☐	☐
3448	Ti'n tecstio at dy ffrindiau yn aml.	☐	☐	☐	☐	☐
2681	I'r canolfan hamdden ti'n mynd.	☐	☐	☐	☐	☐
8680	I'r canolfan hamdden wyt ti'n mynd.	☐	☐	☐	☐	☐
8542	Dwyt ti ddim yn cael mynd adra eto.	☐	☐	☐	☐	☐
6479	Wyt ti'n bwyta cinio rwan?	☐	☐	☐	☐	☐
0283	Ti'n siarad efo fy athro i wythnos nesaf.	☐	☐	☐	☐	☐
9465	Rwyt ti'n eistedd yn y 'stafell fyw.	☐	☐	☐	☐	☐
4844	Efo pwy ti'n dawnsio?	☐	☐	☐	☐	☐
5184	Mae Rhian yn siarad Almaeneg hefyd.	☐	☐	☐	☐	☐
4820	Pwy ti'n siarad efo?	☐	☐	☐	☐	☐
7449	Rwyt ti'n tecstio at dy ffrindiau yn aml.	☐	☐	☐	☐	☐
6316	Wnest ti neud dy waith cartref di yn dda ddoe.	☐	☐	☐	☐	☐

F Poster for Advertising Judgement Experiment

Dach chi'n siarad Cymraeg yn frodorol? (1/2 awr, £5!)

Dw i'n cynnal ymchwil ar Gymraeg llafar ar hyn o bryd. Ar gyfer hyn dw i angen siaradwyr brodorol y Gymraeg i ymuno mewn arbrawf. Mae gan yr arbrawf dwy ran. Yn gyntaf, fasech chi'n gwrando ar gwpl o frawddegau ac wedyn fasech chi'n darllen mwy o frawddegau. Yn y ddwy ran fuasai rhaid i chi ateb ychydig o gwestiynau amdanyn nhw.

Nid oes angen i chi fod yn dda gyda gramadeg neu sillafu neu feddwl bod chi'n siarad "yn dda" – y cyfan sydd ei angen yw i chi fod dros 18 oed a bod yn siaradwr Cymraeg rhugl brodorol!

Dylai'r arbrawf cymryd tua hanner awr neu lai (yn dibynnu ar eich cyflymder) a hefyd ar ôl cwblhau byddech chi'n cael £5.

Os oes gynnoch chi ddiddordeb, cysylltwch â Florian:
elub45@bangor.ac.uk neu ffonio 07932 902 250.

Are you a native Welsh speaker? (30mins, £5!)

I am currently conducting some research on colloquial Welsh. For this I need native Welsh speakers to take part in an experiment in which you will be played some recorded sentences in Welsh and also given a list of sentences which you will then be asked some questions about.

You don't need to be good with grammar or spelling or even think that your Welsh is "good", all you need is to be over 18 years old and a fluent, native Welsh speaker!

The experiment should take about half an hour or less (depending on how fast you are) and on completion you will also be reimbursed £5 for your time.

If you are interested please get in touch with Florian at
elub45@bangor.ac.uk or at 07932 902 250.

G Consent Form for Judgement Experiment

Bangor University's 'Code of Practice for the Assurance of Academic Quality and Standards of Research Programmes' (Code 03)
https://www.bangor.ac.uk/ar/main/regulations/home.htm

COLLEGE OF ARTS & HUMANITIES

Participant Consent Form

Researcher's name: Florian Breit

E-Mail: elub45@bangor.ac.uk
Phone: 07932 902 250

The researcher named above has briefed me to my satisfaction on the research for which I have volunteered. I have been informed that the researcher intends to use the data collected for a dissertation submitted to the School of Linguistics & English Language at Bangor University and in the potential publication of an article in an academic journal. I have also been informed that the researcher intends to make the entire set of data collected during this research, in anonymised and non-identifying form, publicly available. I understand that I have the right to withdraw from the research at any point without any explanation by alerting the researcher of this and that any data collected from me will subsequently be destroyed in this case. I also understand that my rights to anonymity and confidentiality will be respected.

Signature of participant .....

Date .....

This form will be produced in duplicate. One copy should be retained by the participant and the other by the researcher.