Error-Centric Annotation of Learner Corpora

Magdalena Leshtanska Aleksandar Dimitrov

April 9, 2009

Contents

Abstract

This paper presents an annotation method for errors in learner language that strives to be as compact and precise as possible, without sacrificing on generality or correctness. We develop contextualisation and underspecification methods for error annotations, based on a flexible annotation-driven markup.

1 Introduction

Recently, there has been a growing interest in learner language corpora and learner errors analysis Nesselhauf (2004). There is a need to identify errors in text produced by second language learners, to make the automatic processing of those texts feasible (ibid.). Also, the potential of learner errors for educational purposes and language acquisition insight has become increasingly recognized (ibid.), hence, many annotation schemes have been devised to identify and classify learner errors, such as Diaz-Negrillo and Fernandez-Dominguez (2006), Nicholls (2003), Hirschmann et al. (2007)

We propose a novel approach to annotating learner errors, based on the idea that an error is an inconsistency with its surroundings. This basic markup approach can be combined with a taxonomy for error classification. We have coupled it with learner error classification system and used the resulting scheme to annotate a subset of the NOCE corpus (Diaz-Negrillo (2007)), a corpus of beginning to intermediate English learners of Spanish.

Section 2 will introduce the general idea behind our annotation scheme and elaborate on more technical details. Section 3 gives a brief overview of real world performance of our method, as far as this is indeed possible. An annotation manual is contained in Appendix A.

2 Error Taxonomy and Annotation Scheme

Devising an exhaustive taxonomy of errors that can appear in natural language seems a daunting task, since science has so far failed to regularize what *is* a valid utterance of a language. Therefore, our annotation scheme strives to do some things only, and do them well.

2.1 Basic Concepts

Our annotation scheme presents three distinct kinds of errors:

- *Grammatical context.* each introduced token enforces constraints on the utterance, or on parts of it. In (1), *remember* requires a gerund, thus the form *swim* is ungrammatical in this context. We call such mistakes **grammar errors**.

  (1) *I remember swim in the river

- *Semantic context.* In (2), though the grammatical structure is correct, the predicate does not fit the contextual information. This is a contextual error.

  (2) *Yesterday I will go to the fair.

- *Spelling errors.* Spelling errors are typically not influenced by contextual information, except for the normative context of a given language's orthographic rules. Punctuation is a special case, since it may also carry semantic information.

2.2 Calling Errors by Name

The classification of errors, as well as establishing an error's scope and influence on the rest of the sentence are the main tasks in the annotation process of learner language. The following section documents our approach to these problems.

2.3 Error Context

Annotated learner corpora serve one primary purpose: categorizing and cataloguing different kinds of errors that may occur in learner language. In order for such data to be maximally useful, the error annotations have to be as general as possible. We strive to improve the quality of the data by only annotating erroneous tokens in a given error tag, thus not erroneously catching 'good' tokens in our error annotation. This, however, comes at a cost: if only the ungrammatical token is identified as an error, its classification is no longer justified from with in the error annotation itself. Consider the following sentence:

(3) Before she came, *She had going to the super market.

Here, the only erroneous token is *going*, yet it is not by itself a wrong word. This is where error context comes into play: since the error annotation could not possibly tag "going" as a "badly formed verb tense", because, by itself there is nothing wrong with it, we add "Before she came" to its **error context** and apply the aforementioned type to the whole unit consisting of erroneous token and error context. This error dependency mechanism allows error annotations to be confined to a minimal space and still be interpretable by themselves.

Another common pitfall for error taxonomies are ambiguous cases where the annotator has to decide between several possible annotations. This may be avoided by advising annotation of every conceived alternative in such cases, creating overlapping markup. Typically, morphological errors, spelling errors, word order errors, and other kinds of errors will form distinct levels that can stack to a cumulative layer of errors on a single token. It is also interesting to note that context/error pairs can form locking formations, where one error's context can be another error, whose context will point to the original error in turn.

(4) *A mobile phones can be very useful.

Here, the determiner and the subject do not agree in number, but it is not clear which one is wrong. There are two possible corrections, which would result in the subject ending up as plural or singular, respectively. Therefore, we tag two errors, a number agreement error on *a* with context *phones*, and one on *phones*, with context *a*. Note that annotating only "a phones" as an agreement error would not account for the two possible target hypotheses. Such a scheme would have to invent a mechanism for defining multiple target hypotheses[1].

2.4 Error Taxonomy

We adopted an error taxonomy to enrich our error identification annotations by additional error classification. Error type is not dependent on the trigger policy, any grammatical error type annotation scheme may be used here. To adjust an annotation scheme to the error context paradigm, one must simply divide the types among our three basic error type categories: Spelling, Grammar and Context, as already described in 2.1.

For our test taxonomy, we used the taxonomy described in Diaz-Negrillo (2007) as a base, and enhanced it by several concepts from the CLC annotation scheme, as specified in Nicholls (2003). While we retained the general ideas and structure of the aforementioned schemata, we aimed to improve their generality, and systematization.

We chose to follow a hierarchical setup, with some emphasis on typical decision paths an annotator will have to make during the annotation process. The hierarchical nature of our taxonomy made it a good fit for XML schemas, which define relationships in similar ways.

In this taxonomy, ”Error” is the root of all types of errors, with more and more specific error classifications percolating down the tree. An annotator may choose a less specific category in case they are unsure about a certain item. Some errors can indeed not be classified at all, and will have to remain annotated as the general category ”Error.” Since annotation of learner language can be tricky at times, we chose to allow these kinds of underspecification of error class in order to give an annotator the possibility to express an error type more flexibly.

We also chose to alter a few concepts of the error taxonomies we based our own on, specifically, we chose to eliminate certain kinds of error types to make the annotation task less involved and more accurate.

- *Normative Errors.* Some normative concepts are present in almost all European languages, such as capitalization at the be-

[1] Although it is not clear whether our approach would rid us of the necessity of such mechanisms. Word order mistakes pose a significant problem to a one-target-hypothesis-per-error approach.

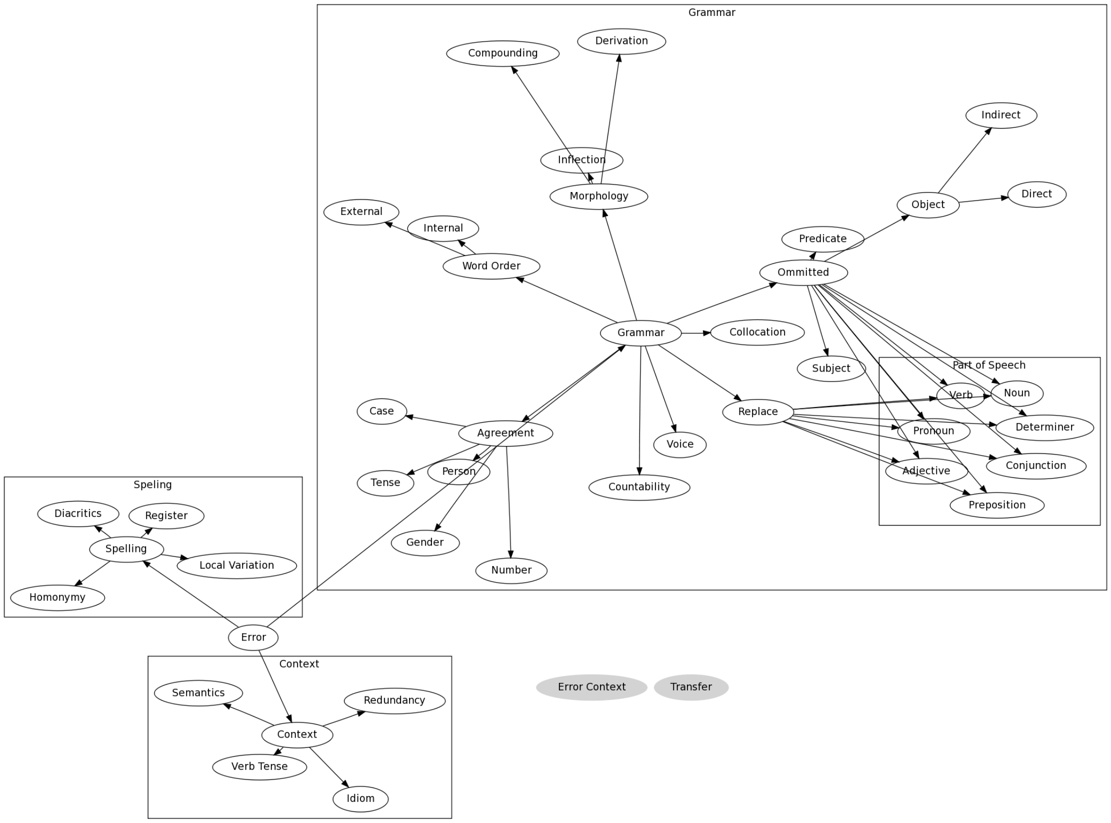

Figure 1: Error classification taxonomy.

ginning of a sentence. We chose to ignore such errors, since they are not a sign of weakness in the target language but likely pure by chance mistakes.

- *Style Errors.* Style is too soft a concept. Errors in style are notoriously hard to peg and reliably quantify. Moreover, our corpus contained data mostly from beginning English learners, where style errors are not as important, evident or relevant.
- *Punctuation.* Punctuation mistakes are a very delicate concern: they can have tremendous effects on any given token string on both a syntactic and semantic level. They might change constituent boundaries, sentence boundaries, argument structures and many more things. Moreover, they tend to have cascading effects on the correctness of a given sentence. Again, because we were annotating a beginner's corpus, we chose to ignore these kinds of errors, since in beginner language punctuation often degrades to line noise very quickly.

We also included a "transfer" attribute to indicate that an error is an L_1 transfer error. This attribute was, however, not used in the annotation process, since the annotators' knowledge of the Spanish language was insufficient to make reliable judgements about it.

2.5 Markup

The context mechanism in our annotation scheme can increase the complexity of the annotation process drastically. Since our annotation technique requires us to build a non-hierarchical graph structure over our input[2], a traditional sequential markup scheme scheme that operates directly on the text or spans over a piece of text will run into at least two kinds of problems:

- *Interleaving annotations* occur when an error doesn't end before another one begins. Given the tokens $\tau_1\tau_2\tau_3$, and two errors η_1 and η_2 ranging over tokens τ_1, τ_2 and τ_2 τ_3 respectively, the resulting markup will be confusing or outright impossible to read: $(\eta_1)\tau_1(\eta_2)\tau_2(/\eta_1)\tau_3(/\eta_2)$. This is particularly a problem with XML, since the specification[3]explicitly disallows interleaving markup.

[2] Relating words and their dependencies more like a net than as a hierarchically nested span over a character sequence

- *Greedy annotation* covers tokens entirely uninvolved in the *"production"* of an error. If of the token sequence $\tau_1\tau_2\tau_3$ only τ_1 and τ_3 are erroneous, marking the entire token sequence with an error annotation would falsely accuse the otherwise completely innocent token τ_2.

Based on these assumptions, we decided to decouple the *error markup* from the *corpus data.* Specifically, to our annotation method, the errors and the text are two entirely different data structures. Every particular error can reference tokens within the corpus using a **key**, similar to the way modern Relational Data Bases reference their data. This makes the index more accessible and easier to maintain and eliminates both problems above, because every single annotation can be completely independent from all other annotations. The source files for our corpus consisted of plain text files from the NOCE-corpus of beginning to intermediate Spanish learners of English. Our corpus format is defined in 2.1.

Definition 2.1. *A corpus $\mathcal{C} = \langle \mathcal{T}, \mathcal{E} \rangle$ is defined as a pair of a set of tokens, $\mathcal{T}$ and a set of errors, $\mathcal{E}$. Every token τ_i is indexed with a unique identifier subscript i*[4]*. An error annotation is a tuple $\eta = \langle E, C, \theta, t, c \rangle$, where*

- *$E \subseteq \mathcal{T}$ is a nonempty set of indices of erroneous tokens*
- *$C \subseteq \mathcal{T}$ is a possibly empty set of context tokens*
- *θ is the type of the error,*
- *s a string denoting an optional target hypothesis hint, and*
- *c an optional comment.*

The error type θ is defined as an ordered sequence of categories from the taxonomy presented in 2.2. Furthermore, $E \cap C = \emptyset$ for all $\eta \in \mathcal{E}$.

This general data structure is translated to XML. A DTD ensures correct usage of the markup. The authors also designed a graphical user interface based annotation tool[5]which hides the implementation details of the corpus data from the user and enables a rapid annotation

[3] Located at `http://www.w3.org/TR/REC-xml/`

[4] Note that the identifier may not consist entirely of numbers, since it's XML type is `ID`, which demands identifiers to be alphanumeric sequences. Furthermore, the identifiers need not adhere to any particular order, as long as they are unique.

process. The user marks tokens from the corpus as erroneous, assigns them a type, possibly an error context and suggest a target hypothesis or records a comment entirely via the interface.

2.6 Limitations

Word order errors still pose a particular complication even to a non-sequential scheme such as ours. We are not able to account for the complex dependencies occurring in some word order mistakes simply by a dependency-graph.

Since our minimal atomic unit is a token, we depend on a tokenization process. Furthermore, we cannot account for mistakes that occur on a scale smaller than a token.

3 Assessment

Statistical inter-annotator agreement measures are a common quality assessment method used in corpus linguistics and related fields. Hereby, annotations made on a particular data set by two or more annotators are compared using quantitative methods. Artstein and Poesio (2008) give an overview of currently employed methods.

While inter-annotator agreement measures have been applied successfully to various corpus linguistic tasks, so far they have not found wide usage among learner language annotation. We believe the current techniques may not applicable to this particular problem domain.

3.1 Unitization and Multidimensional Markup

Existing inter-annotator agreement measures all assume the presence of atomic units in the corpus data, which are annotated by the annotators of a certain data set. The annotations *over these units* are then used to calculate an agreement coefficient. However, constituents of learner language, being fairly diverse in nature, are not as easily contained in atomic units. Instead, a common task in the annotation of learner language is *delineating* the extent of a certain mistake.

[5] The tool is implemented in Haskell (`www.haskell.org`) on top of the GTK framework (`www.gtk.org`). It free and open source. The sources to the tool and the XML markup format are given in appendix C.

Artstein and Poesio (2008) briefly discuss unitization and go on to note that it has thus far not been exhaustively researched. Even more importantly, they explicitly comment on the unknown status of the validity of the only inter-annotator agreement measure in the corpus linguistic literature, α_U, presented in Krippendorff (1995). Apart from being untested, α_U also seems to have problems with overlapping markup both in the text, and between annotation sets (both of which are frequent in learner language data.) The measure also assumes annotation spans to be continuous, which is not the case in our data.

3.2 Quantificational Analysis

Facing these theoretical difficulties, we reached the conclusion that the only viable way to define inter-annotator agreement over learner language data we could improvise[6]would still not yield interpretable results. In order to quantify our analysis efforts, we analyzed the two annotation sets with respect to their annotation's intersections.

Table 1 shows the total amount of corresponding data in the individual markup. The table's labeling reads as follows: s stands for a non-empty intersection between the two data sets (or prefix relation for the error types) and i for equality. o, y, r denote the data type: o stands for error tOkens, y for the error's tYpe, and r for the errors context (or *tRigger*). Thus, $so \wedge sy \wedge ir$ is the number of all annotations that have a partial overlap on the error tokens E, a prefix match on the error type θ, and identical error context.

4 Conclusion

4.1 Possible Extensions

After assessing the quality of our data, we reached the conclusion that the error format described in 2.2 might benefit from several re-

[6] It would in theory be possible to regard tokens as units, and that is indeed how our scheme is currently implemented. An agreement measure would then interpret annotations locally on these units only. However, this method has two major drawbacks: firstly, it does not account for more than one annotation on a given unit, which happens frequently. Also, if one annotator marks a set of tokens as an error consisting of more than one token, and the other annotator marks these tokens with several errors of the same category (which can happen, for example, with word order mistakes, as well as complex agreement mistakes), this method would fail to account for the discrepancy.

1 Total:	(905,933)	(98.69%,101.74%)
$io \wedge iy \wedge ir$:	438	47.76 %
$so \wedge iy \wedge ir$:	448	48.85 %
$io \wedge sy \wedge ir$:	447	48.74 %
$io \wedge iy \wedge sr$:	480	52.34 %
$so \wedge sy \wedge ir$:	476	51.90 %
$io \wedge sy \wedge sr$:	503	54.85 %
$so \wedge iy \wedge sr$:	499	54.41 %
$so \wedge sy \wedge sr$:	544	59.32 %
$io \wedge iy$:	486	52.99 %
$so \wedge iy$:	506	55.17 %
$io \wedge sy$:	509	55.50 %
$so \wedge sy$:	551	60.08 %
$io \wedge ir$:	525	57.25 %
$so \wedge ir$:	578	63.03 %
$io \wedge sr$:	623	67.93 %
$so \wedge sr$:	719	78.40 %
io:	629	68.59 %
so:	730	79.60 %

Table 1: Absolute amount of annotation overlap.

finements. Adding a field for part of speech tags might contribute to the clarity of the data, as well as to its searchability. The annotated corpus could be queried for erroneously placed verbs or prepositions, for example.

Moreover, we came to the conclusion that defining the error context as a set of tokens might be misleading or at least difficult to understand in case the error context does not constitute one sequence, but several scattered sequences, such as proper nouns or syntactic constituents. C could therefore be turned into a set of sequences of token indices.

Annotating a corpus with this schema is quite laborious, and borders on impossibility without proper support from an annotation tool, such as the one we had to devise. During the process, however, our tool was constantly improved according to the annotators' ideas, and made the annotation process easier in the process. Partial automation and other features might increase annotation comfort even further.

The taxonomy presented in proved to be a little unwieldy. In particular, it did not clearly distinguish between annotating a target and annotating an error type. The next subsection proposes a refinement that might make the annotation process more precise.

A part of speech tagger could aid the annotators and enhance the corpus data significantly. However, there are only a few reports on reliable part of speech tagging for learner language. Hirschmann et al. (2007) presents one such approach.

4.1.1 Towards an underspecification formalism for target hypotheses

During the annotation process, we discovered that our taxonomy branches for *omission*, *replacement*, and *redundancy* could be turned into a stub of a formalism for underspecification of target hypotheses. Instead of giving a string for a target hypothesis, such a formalism would make it possible to approximate the target and therefore allow for more flexibility in the markup. Note, however, that these forms of omission, redundancy and replacement differ from the ones included in the error taxonomy.

The taxonomy tries to account for *what is wrong* with a given string of text. A target hypothesis would try to make assumptions about *how this could be fixed.* Our categories in the error taxonomy suggesting manipulation of the input text were explicitly designed to catch cases where a clear reason for an error could not be found, and the syntactic environment of a given set of tokens would require the text to be changed entirely. Such subcategorization mistakes could be granted their own category and the target hypothesis could account for the necessary steps in order to ensure grammaticality.

This would also allow for existing annotations to be combined with *generic instructions for correcting the input* and increase the granularity of the data.

A Annotation Manual

At any point, if you advance further down the taxonomy tree through a decision you make later on, do not mark the error as the more general category, but mark it as the more special category instead.

A.1 Steps in the annotation process

1. Look at each token in turn, assuming all the rest is grammatical.
    - If the token is not spelled correctly, record a spelling error.
    - If an erroneous token τ_1 is erroneous because it does not harmonize with a token τ_2 from its context, mark τ_1 as an error and τ_2 as trigger.
    - If the utterance is ungrammatical, i.e. it does not constitute a part of the language, mark it as a grammar mistake.
    - If the sentence is utterable under a certain condition, but this condition is not met here, (i.e. forbidden by the general context of the text) mark it as a contextual error.
    - Now, choose a subcategory from the branch you have chosen. Refer to the graph in 2.4. Try to be as specific as you can, otherwise, if not completely sure, choose the upper node. Try to see in what way a token can be changed, as to fit all the additional context. To find out which branch to choose, follow the instructions under B
    - Repeat the process until all possible errors have been annotated, then move to the next token.
2. If the context is too erroneous and this erroneous context part affects directly the immediate context of your token, try to find another solution, because otherwise you will be building error hypotheses on errors. If it doesn't (say, it is a spelling mistake),

consider it as though it was correct while looking for a directly interacting token as trigger.

3. *Local markup*: Keep it as local as possible, try to mark as triggers tokens that are grammatically dependent on each other in normal language production.
4. Go from tokens up to other units, such as phrases, and be minimalistic, unless this makes you lose important information.
5. Sometimes the error and/or the trigger will be a sequence. Take the minimal unit you can that doesn't lead to loss of information. Always choose the head of a constituent or compound word when it isn't the whole constituent that plays a role in error formation.

   (5) The lovely, charming and amazing lady *laugh* a lot

   Here, only take "lady" as a trigger. But in cases like (6), your trigger should be "to study", and not only "to", because otherwise you'd lose information. Maybe the learner doesn't have a problem with prepositions, but with infinitives.

   (6) I go for to study in Granada.

6. Avoid cascading and annotating errors based on corrected errors. Only look at the unaltered text.
7. Annotate every possibility. Since we do not apply cascading rules, one can use weighting algorithms to prefer one error to another in the case of concurrent errors:

   (7) a. *A* mobile **phones** can be very useful.
   b. **A** mobile *phones* can be very useful.

8. Only correct when necessary, do not be tempted to rephrase a construction, just because it would sound better or is more common, if there is nothing explicitly wrong with it.
9. No style errors.
10. No punctuation errors
11. No capitalisation, only if it is something specific to English (Spelling→Register), such as, say, capitalisation of proper nouns or months.

12. If you have to choose between extremely similar possibilities, choose the one whose target hypothesis would have the least impact on the text in terms of phrasal constituents.

B Branches in the taxonomy

B.1 Collocation vs. Replace/Omission/Redundant

Verb and preposition collocations (i.e. where a certain verb forces a certain collocation) are to be marked as Collocations, not as an erroneous word. Words that occur with plural (*one of*, *between*) are also collocation bound, and an error in agreement is thus a collocation error.

B.2 Gerunds

Gerunds are nominalized noun forms. Place mistakes related to the formation of gerunds in Morphology → Inflection, and errors related to the tense in which a gerund ought (not) to occur in Grammar → Verbtense.

See also: Derivational Mistakes

B.3 Morphology

Morphological negation and plural formation mistakes often occur in English learner language. We tag them as Morphology and Morphology → Inflection instead of choosing a more specific category, but might extend our scheme in the future.

B.4 Derivational Mistakes

If a word is wrongly derived from another, (nominalization, suffixation, prefixation, etc. . .) tag it as a morphological derivation mistake.

B.5 Animate/Inanimate

Is not covered by the scheme right now. Use Replace → Pronoun or similar. Might be added in the future.

B.6 Subject, Object, Predicate vs Pronoun, noun, verb.

If a whole constituent is missing or wrong, use the constituent classes. If it's one word out of the constituent, use the part of speech classes. Again: try to be as general as possible.

B.7 Number Agreement vs. Redundancy

(8) a dogs

a is redundant, but because of number agreement. So you must annotate both, creating a circular dependency, where an erroneous *dogs* has as error context *a* and an erroneous *a* has as context *docs*

B.8 Compound nouns

If a compound noun of two or more words is erroneous, annotate only the head noun, carrying the mistake.

B.9 Sequences

For both triggers and erroneous tokens, it is best to use a single token. In case of a phrase, only take the keyword and at most any relevant information bearing token. In such a case, do not care about order.

Still, fixed expressions, such as Collocations or Idiom can be counted as one unit. In such a case, annotate the whole phrase, even if it contains erroneous tokens. Only do not annotate the target if it is within the idiom.

B.10 Word order

- *Internal word order.* Words should be shuffled within the marked sequence.

  (9) Not only I must study → Not only must I study

- *External word order.* Specified word needs to be moved somewhere else, its elements moving together as a unit.

  (10) I the river like → I like the river

B.11 Verb Tense

If the verb tense is wrongly formed, (*he had having a nice time*), it's a grammatical error. Also, if there is a typical trigger word for verb tenses (such as *since*, *before*, *if*) and the tense is wrong, it's a grammatical error. If the tense doesn't fit because of the broader context, it's a contextual mistake.

C Source Code and XML Data

The complete code, as well as an up-to-date version of this paper can be found online at `www.github.com/adimit/ll-annotation`. The annotation schema's DTD is `AnnotationScheme.dtd` in the root directory of the repository.

References

Ron Artstein and Massimo Poesio. Inter-coder agreement for computational linguistics. *Comput. Linguist.*, 34(4):555–596, 2008. ISSN 0891-2017. doi: http://dx.doi.org/10.1162/coli.07-034-R2.

Diaz-Negrillo. *A Fine-Grained Error Tagger for Learner Corpora.* PhD thesis, University of Jaen, Spain, 2007.

Diaz-Negrillo and Fernandez-Dominguez. Error tagging systems for learner corpora. *Revista Espanola de Linguistica Aplicada (RESLA)*, 19:83–102, 2006.

Hirschmann, Doolittle, and Ludeling. Syntactic annotation of non-canonical linguistic structures. In *Corpus Linguistics 2007*, 2007.

Krippendorff. On the reliability of unitizing contigious data. *Sociological Methodology*, 25, 1995.

Nesselhauf. *Learner corpora: Learner corpora and their potential for language teaching*, pages 125–152. Studies in Corpus Linguistics 12. John Benjamins, 2004.

Nicholls. The cambridge learner corpus - error coding and analysis for lexicography and elt. In *Proceedings of the Corpus Linguistics 2003 Conference (CL 2003).*, 2003.