Error-Centric Annotation of Learner Corpora

Magdalena Leshtanska Aleksandar Dimitrov

April 9, 2009

Contents

I am an abstract!

1 Introduction

Literature overview. Bla Bla.

All of the annotation schemes mentioned above focus on the **text** itself, and will often even advise the annotator to *modify* the underlying text with the annotation. We believe that a more error-centric annotation of the data can be beneficial.

Firstly, every annotation scheme that operates directly on the text or spans over a piece of text will run into 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[1]explicitly disallows interleaving markup.
- *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.

2 Error Taxonomy and Annotation Scheme

In order to identify errors occurring in learner language, one has to first classify them. However, even then, identifying which class a particular error belongs to is in no way an easy job. 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 taxonomy will try to do some things only, and do them well. In particular, we will not cover punctuation mistakes, since their possible corrections can have cascading effects on the rest of the sentences and suddenly give rise to all kinds of new errors.

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

(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[2].

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 **?** as a base, and enhanced it by several concepts from the CLC annotation scheme, as specified in **?**. 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.

[2]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.

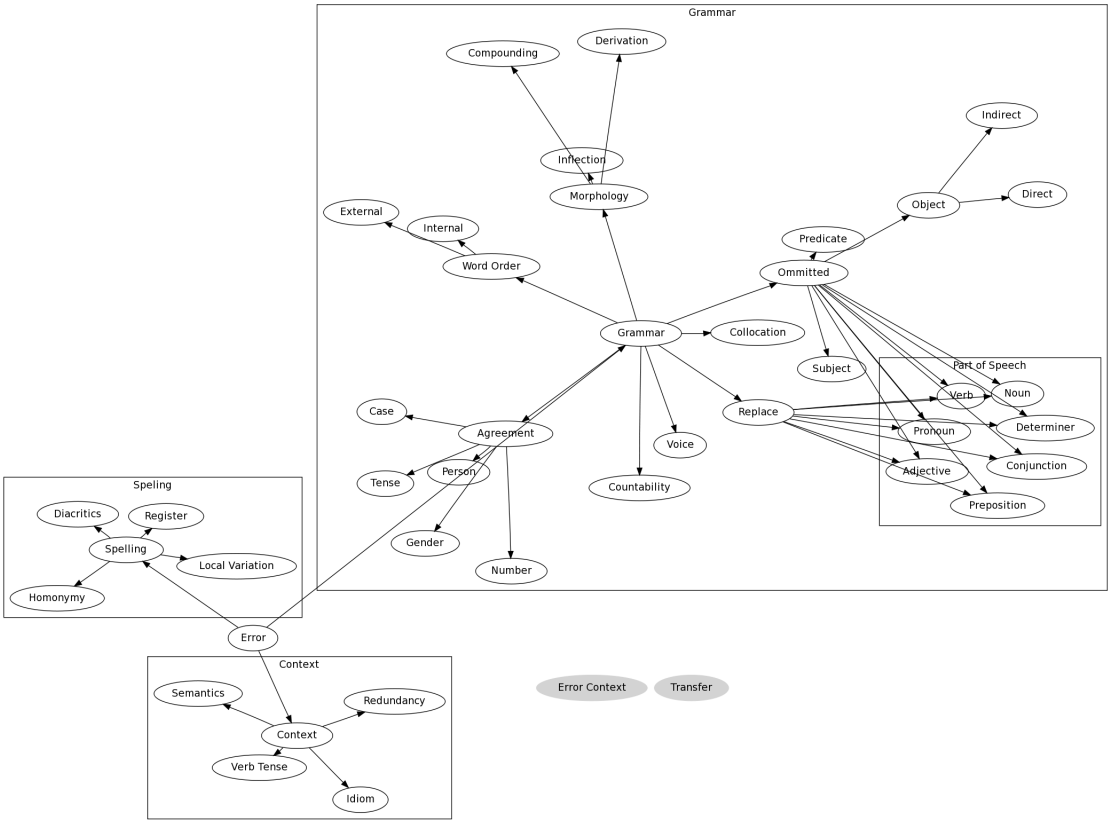

Figure 1: Error classification taxonomy.

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 beginning 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, the NOCE 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 source files to 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[3]. 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.*

[3]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.

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[4]which hides the implementation details of the corpus data from the user and enables a rapid annotation 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

Some elements, for example word order errors have enforced context, but it is hard to express via our schema, so we don't do it. No correct positions specified by word order errors - also hard to express. One could assign them a target position after an existing token, but then one must also keep track of other "reorderings" by multiple word order errors - I would like beer some to drink - here, both erroneous units - some and to drink should be placed between like and beer, but one must take care to order them correctly. Also, as our schema tags all possible errors, in the case of interchangeable errors (cf XXX), one cannot be certain if the error annotated is really the one, made by the learner, and not the other possible one. Most importantly, there are sometimes more than one places where a word sequence can fit in.

Errors within a token cannot be analyzed – for example, agglutination, because a token is our smallest unit.

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. **?** 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.

[4]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 **??**.

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.

? 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 **?**. 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[5]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.

[5]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.

4 Conclusion

4.1 Possible Extensions

After assessing the quality of our data, we reached the conclusion that the error format described in **??** might benefit from several refinements. 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. **?** 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.