Error-Centric Annotation of Learner Corpora

Magdalena Leshtanska Aleksandar Dimitrov

April 8, 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 $a_1 a_2 a_3$, and two errors e_1 and e_2 ranging over tokens a_1, a_2 and a_2 a_3 respectively, the resulting markup will be confusing or outright impossible to read: $(e_1)a_1(e_2)a_2(/e_1)a_3(/e_2)$. 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 tokens $a_1 a_2 a_3$ only a_1 and a_3 are involved in the production of an error (say, an agreement error) marking the entire token sequence as erroneous would involve blaming the otherwise completely innocent token a_2. This can significantly increase the noise in a corpus.

There is little data about the primary use cases for learner corpora known to us, but it does not seem too far fetched to assume most linguists interested in such data are not likely to browse through the corpora in order to find an amusingly written piece of text. Instead, it seems to be safe to assume that most would be interested in the **kinds of errors** that can occur in such a corpus, and their particular properties. The error data should thus contain as little noise as possible and be maximally specific while retaining a large amount of generality. It should furthermore be easily accessible, without having to actually read the texts within the corpus itself.

Based on these assumptions, we decided to decouple the *error markup* from the *corpus data*. Specifically, to our annotation scheme 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.

2 Error Taxonomy and Annotation Scheme

Learner errors are not easily classified due to their wide variety of appearance, kind, and ambiguities involved. Typically, an annotation scheme will strive to cover as large a scope it possibly can unambiguously classify, yet there are many practical limitations to the potential coverage one can achieve.

One of the most prevalent restrictions is the need to devise the annotation manual with great attention to detail, and to perform a meaningful assessment of the quality of the annotated data after the annotation process. Even a most meticulously well-defined manual is still going to be read and applied by humans, who will have their personal intuition. It is easy to see why this inherent individuality poses problems to the consistency of any kind of

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

annotated data. A well-written annotation manual will limit the effect this natural variance can have on the data.

2.1 Calling Errors by Name

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.

Our annotation scheme presents three distinct kinds of errors.

1. *Spelling errors* are a primitive form of error. They contain no metadata or any kind of further specification except maybe a target word they should have ended up as.
2. *Context errors* are otherwise grammatical sentences which are, however, semantically incorrect given the current context.
3. *Grammar errors* simply do not belong to the target language because of their erroneous morphology or syntax.

[Follows information about triggers, and how the play into this general scheme]

2.2 Representing Errors in Annotations

An error annotation is a tuple $\langle E, C, \theta, t, c \rangle$, where

- E is a nonempty set of indices of erroneous tokens,
- C 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 2.1.

We hope this general format to be sufficient for annotation within our current goals, and extensible enough to cover future refinements of the annotation standard, such as the addition of part of speech tags or syntactic markup. Keeping the error annotation in a separate data structure, allows for the annotation itself to be more detailed and flexible.

3 Annotation Manual

3.1 Spelling Errors

A spelling error is an error that *cannot* be explained by mistakes in morphology, e.g. derivation or inflection. It will typically comprise only one word. If it comprises more than one word, it is seen as a *compound* spelling error, i.e. two tokens separated by a space should have been concatenated to yield only one token. Similarly, if the target hypothesis accounts for more than one word, the token should have been split up into two.

3.2 Context Errors

Context errors occur in otherwise completely grammatical sentences, which, however do not convey what was obviously intended by the author taking context into account. Lexical errors are also context errors, if the given word is grammatically valid within the sentence.

3.3 Grammatical Errors

3.4 Devising Target Hypotheses

Retain existing input by the author of the text to the greatest extent possible. The text should be only minimally altered in the target hypothesis.

4 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. In fact, we believe the current techniques are not applicable to this particular problem domain.

4.1 Unitization and Multidimensional Markup

Existing inter-annotator agreement measures the presence of atomic units in the corpus data, which are annotated by all annotators of a certain data set. This is the case with part of speech tagging or syntactic classification of chunked sentences, where all annotators are presented with a set of tokens or a set of chunks, respectively, which they have to annotate.

The annotations *over these units* are then used to calculate an agreement coefficient that will usually range from -1 to 1. However, learner language is a lot more diverse in nature, and it is not as easily possible to partition it into atomic data units. Instead, annotators have to *find data points* and also *manually delineate* them, according to the annotation manual. This brings the problem of *unitization* into play.

[**?**] 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 has several more problems that make it an ill fit for learner language data. As mentioned in [**?**], Krippendorf's α_U assumes markup to be non-overlapping. This is certainly not the case with learner language, where one error may be nested within in another error[2]. Also, if a segment annotated by one annotator spans more than one segment annotated by the other annotator, α_U will not calculate their agreement correctly.

We see yet another problem with applying α_U to learner corpora: we showed earlier that often learner errors will not present a coherent unit, but may be scattered across several tokens. α_U does not account for the possibility of annotations deviating *within* the boundaries of a marked up error.

4.2 Quantificational Analysis

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 (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.

4.3 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 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

[2]This happens very frequently with spelling errors.

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.

sequences, such as proper nouns or syntactic constituents. C could therefore be turned into a set of sequences of token indices.

4.3.1 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.