Analyzing Learner Language: Towards A Flexible NLP Architecture for Intelligent Language Tutors

Luiz A. Amaral
UMass Amherst
amaral@spanport.umass.edu

Detmar Meurers　Ramon Ziai
Universität Tübingen
{dm,rziai}@sfs.uni-tuebingen.de

SVN Version: 861

Abstract

Intelligent Language Tutoring Systems (ILTS) typically focus on analyzing learner input to diagnose learner errors and provide individualized feedback. Despite a long history of ILTS research (cf. Heift & Schulze, 2007), such systems are virtually absent from real-life foreign language teaching (FLT). Arguably, one reason for this state of affairs is that FLT activity design and its impact on the system architecture are rarely considered in ILTS research.

In this paper, we argue that a demand-driven, annotation-based Natural Language Processing (NLP) architecture is well-suited to handle the demands posed by the heterogeneous learner input which results when supporting a wider range of FLT activity types. We illustrate how the Unstructured Information Management Architecture (UIMA) can be used in an ILTS, thereby connecting the specific needs of activities in foreign language teaching to the current research and development of NLP architectures in general. Making the conceptual issues concrete, we discuss the design and realization of a UIMA-based reimplementation of the NLP in the TAGARELA system, an intelligent web-based tutoring system supporting the teaching and learning of Portuguese.

Contents

1 Introduction

In the context of Computer-Assisted Language Learning, Intelligent Language Tutoring Systems (ILTS) provide individualized feedback to learners working on activities. ILT systems may also individually adjust the sequencing of instruction. Typically the focus of the analysis is on form errors made by the learner, even though in principle feedback can also address aspects of meaning. In addition to diagnosing errors in ill-formed learner language, Intelligent Computer-Assisted Language Learning tools can also be used to highlight form or meaning properties in well-formed sentences to support language awareness.

While for some restricted exercises it is possible to anticipate all potential learner input and intended system responses, for most types of language learning activities such a direct mapping between potential learner input and feedback is not feasible (cf. Nagata, 2009). Instead, it is necessary to abstract from the specific string to more abstract, general classes by automatically analyzing the learner input using algorithms and resources from Natural Language Processing (NLP). Generation of feedback can then be based on the information obtained through such NLP analysis.

Common to most, or possibly all, current ILTS is that the NLP modules are integrated into a pipeline architecture (cf, e.g., Rypa & Feuerman, 1995; Levin & Evans, 1995; Nagata, 2002; Delmonte, 2003; Heift, 2003). The system calls the NLP modules in a pre-defined order, transforming one data structure into another and terminating when specific conditions are met, e.g., when the learner response matches a pre-stored target response, or when spell checking fails.

Such a pipeline architecture works well as long as the system deals with learner input from activity types that are uniform with respect to the required NLP processing. For example, in the E-Tutor (Heift, 2003) and Robo-Sensei (Nagata, 2002), the two only ILTS standardly used in current real-life foreign language teaching, the learner answers consist of single sentences and the lexical material to be used by the learner is constrained explicitly by listing the stems or implicitly by eliciting the student answers through translation (cf., Amaral & Meurers, submitted, sec. 3.1). The NLP diagnosis and feedback generated apparently is the same for all activities.

A uniform pipeline architecture becomes problematic, however, when trying to integrate a wider range of activity types resulting in learner input of a heterogeneous nature, which potentially should also be evaluated using various criteria. Based on our experience from creating a range of activities for TAGARELA (`http://purl.org/icall/tagarela`), a web-based ILTS for the instruction of Portuguese as a foreign language, in this paper we thus argue for a more flexible, demand-driven architecture for this type of systems.

In such an architecture, the use and sequencing of the different NLP modules is triggered by demands for particular information based on the activity models for the different activity types. Each NLP module enriches the input with annotations until all information required to evaluate the learner's performance to provide feedback on a particular activity is present. In other words, the fixed algorithmic pipeline is replaced by whatever processing sequence is needed to obtain the particular information that is required by the activity model to provide feedback for a given exercise.

Relating this point to the broader NLP context, an ILTS can be viewed as an instance of an application required to deal with heterogeneous input and different information needs based on that input. Our approach thus is reminiscent of current approaches to information extraction, where, e.g., IBM's OmniFind makes use of the Unstructured Information Management Architecture (UIMA, Ferrucci & Lally, 2004) to obtain a range of annotations depending on the specific information needs.

In this paper, we show how such an architecture can be realized for an ILTS. We discuss our reimplementation of the NLP in the TAGARELA system based on the UIMA architecture and showcase the benefits of this demand-driven, annotation-based architecture.

2 TAGARELA and the Analysis it Needs to Perform

2.1 The NLP modules and what they are used for

The TAGARELA system makes use of a number of NLP modules. The form analysis includes a tokenizer which takes into account specifics of Portuguese such as cliticization, contractions, and abbreviations. Full-form lexical lookup returns all analyses based on the CURUPIRA lexicon (Martins et al., 2006), which generally provides multiple analyses for each token. Finite state disambiguation rules are used to narrow down this lexical information based on where in a sentence the token appears, in the spirit of Constraint Grammar (Karlsson et al., 1995; Bick, 2000, 2004). A bottom-up chart parser is used to check agreement, case and some global well-formedness conditions. To analyze the meaning of the learner input, shallow semantic matching strategies between the student's input and target answers are used, in line with the Content Assessment Module proposed in Bailey & Meurers (2008).

2.2 Analyzing Portuguese learner data

To identify the types of errors the system has to handle, we collected a corpus of approximately 10,000 words from written assignments of students in an introductory Portuguese course at the college level, and we created a taxonomy of expected errors. Among the most common classes of errors in our corpus were *spelling* (24%), *agreement* (16%), *missing word* (12%), *extra word* (7.5%), and *word choice* (3.2%). Overall, the error taxonomy used by TAGARELA consists of seven general groups of errors: *non-words*, *subcategorization*, *agreement*, *missing word*, *extra word*, *word order*, and *word choice*. Some of the error groups are subdivided further; for example, *agreement* is divided into *subject-verb* for person and number, and *adjective-noun*, *determiner-noun* and *subject-predicative* for number and gender.

The most common *agreement errors* we observed are between determiners and nouns, such as the gender agreement error in (1), followed subject-verb agreement errors in person or number as illustrated by (2). TAGARELA's parser is used to identify such errors.

(1) Eu vou **na** cinema.
I go to the$_{fem}$ cinema$_{masc}$

(2) Eu **trabalha** no journal.
I$_{1.sg}$ work$_{3.sg}$ at the newspaper

The system uses the shallow semantic matching modules to deal with errors classified as *missing words*, *extra words*, and *word choice*. Missing words range from typical function words, such as the prepositions in (3), to lexical heads, such as the missing verb in (4); in the examples, the missing word is shown in bold in square brackets.

(3) Nós começamos **[a]** falar com eles.
we started to speak to them

(4) Eu **[tenho]** muito trabalho. Tchau! Obrigada!
I have much work good bye thank you

The most common cases of *extra words* were extra articles, such as in (5). We also found cases of extra prepositions, complementizers, and pronouns, such as the clitic 'se' in (6); in the examples, the extra word is shown in bold.

(5) Vocês **os** dois sempre querem a sobremesa.
you the two always want the desert
'Both of you always want desert.'

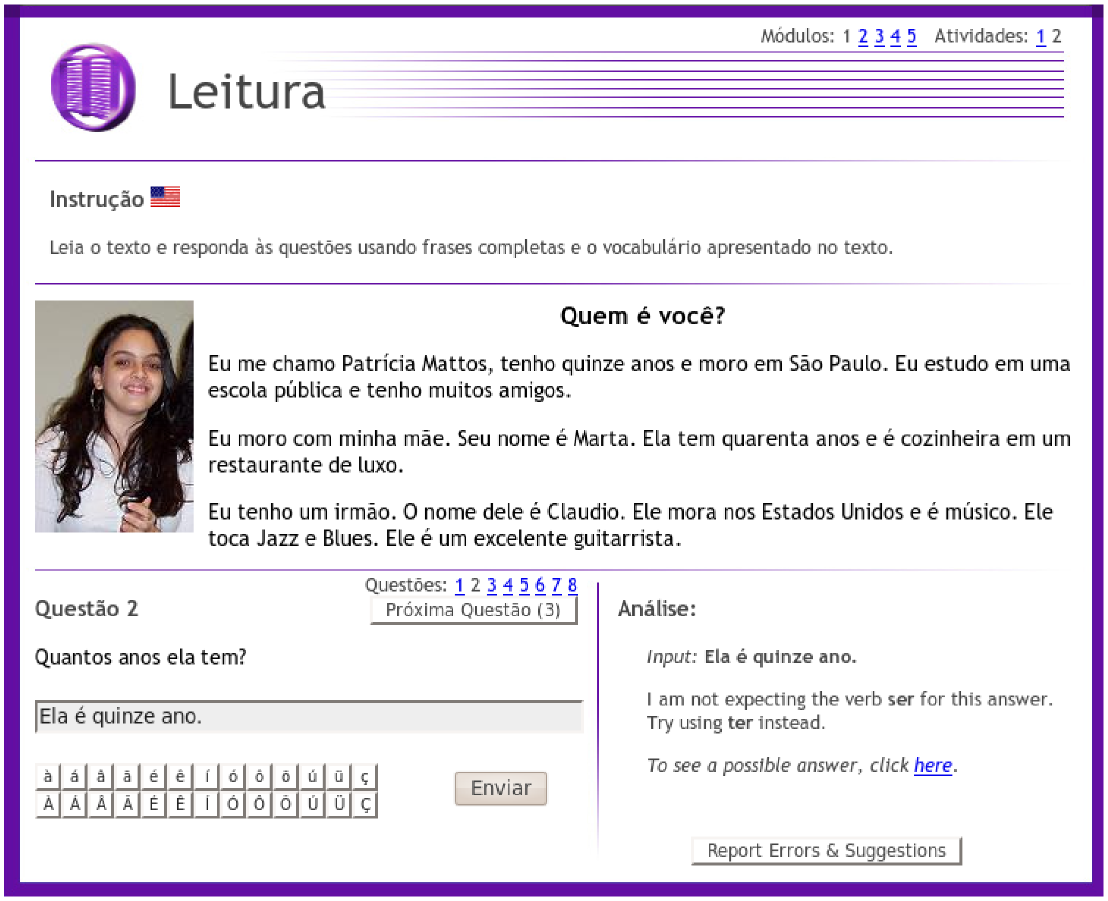
(6) Eu me chamo-**se** John.
I myself call-oneself John
'My name is John.'

Word choice are errors that have their origin in false cognates or in false translations chosen from bilingual dictionaries. In example (7), the student translated 'have a drink' literally into '*ter* uma bebida', even though in Portuguese the correct expression is '*tomar* uma bebida' (to take a drink).

(7) Eu pretendo ir ao clube e **ter** uma bebida.
I intend to go to the club and have a drink

2.3 Providing feedback

Once errors are diagnosed in the student input, a module called Feedback Manager decides on the error message to report, generates the message, and displays it to the student. An example of a feedback message can be seen in Figure 1, where the student made a word choice error, among others. The feedback message contrasts the infinitive of the word used by the learner with the infinitival form of the correct word choice.


Módulos: 1 2 3 4 5 Atividades: 1 2
Próxima Questão (3)
à á â ã é ê í ó ô õ ú ü ç
À Á Â Ã É Ê Í Ó Ô Õ Ú Ü Ç
Enviar
Report Errors & Suggestions

Leitura

Instrução

Leia o texto e responda às questões usando frases completas e o vocabulário apresentado no texto.

Quem é você?

Eu me chamo Patrícia Mattos, tenho quinze anos e moro em São Paulo. Eu estudo em uma escola pública e tenho muitos amigos.

Eu moro com minha mãe. Seu nome é Marta. Ela tem quarenta anos e é cozinheira em um restaurante de luxo.

Eu tenho um irmão. O nome dele é Claudio. Ele mora nos Estados Unidos e é músico. Ele toca Jazz e Blues. Ele é um excelente guitarrista.

Questões: 1 2 3 4 5 6 7 8

Questão 2

Quantos anos ela tem?

Ela é quinze ano.

Análise:

Input: **Ela é quinze ano.**

I am not expecting the verb **ser** for this answer. Try using **ter** instead.

To see a possible answer, click here.

Figure 1: Example feedback provided by TAGARELA

3 Demands on the NLP architecture

3.1 Handling a range of activity types

We mentioned in section 1 that one of the motivations for having a demand-driven architecture in an ICALL system is to adjust the processing of the input and the feedback massages to different types of activities. In this section we describe the different types of activities supported by the TAGARELA system, and present some of their specifications. In section 4.2, we illustrate how TAGARELA uses that information to process the student input and provide feedback.

TAGARELA includes six types of activities for beginning learners of Portuguese at the university level: reading, listening, description, rephrasing, vocabulary, and fill-in-the-blanks. The activity types represent different tasks that have to be performed by the learner. Activity specifications directly affect a) the nature of the student input, b) the type of NLP processing that is necessary to handle such input, and c) the nature of the feedback message that should be generated.

3.1.1 Learner input properties

Reading, listening, description and rephrasing require the learner to produce a full sentence. The target answer for vocabulary activities is usually a noun phrase, while the fill-in-the-blank exercises are typically answered with one word per blank.

3.1.2 Processing requirements

Because of the different expected input types, the NLP modules required by each activity can vary. Fill-in-the-blanks only require the spell-checker and a simple matching mechanism, while for all other types of activities the input processing usually starts with tokenization and lexical look-up, and may end up requiring a full syntactic analysis of the input sentence. Even within the same type of activity, the properties that need to be identified by the NLP modules can differ: some reading exercises target forms explicitly given in the text, others require more semantic analysis or inferences.

To give a concrete example, reading comprehension questions in TAGARELA all take full sentences as answers, yet are heterogeneous in terms of processing and feedback. In the most simple reading comprehension task based on *text identity*, the student is required to identify an answer explicitly given in the text. For such a task, content analysis can often be successfully performed using simple string matching. In the second type, corresponding to a more general *information extraction* task, the student is required to extract information which is given in the text but not as a contiguous sequence. While full string matching with simple edit distance measures will not be sufficient for such tasks, shallow content analysis using token matching can still be successful. Finally, for reading comprehension tasks requiring the student to draw inferences based on a text, content analysis requires deeper analysis to compare concepts and relations in the learner answer with those in the target answer.

Besides the need for specific NLP analysis for different activity types, there is also the issue of the reliability of such analysis. For example, the output of some matching techniques can be more reliably used to generate feedback messages with activities such as rephrasing and description than with activities like reading and listening. This happens because the elicitation techniques used by activities like rephrasing and description more effectively constrain the possible variations in student's response than the *wh*-questions used in reading and listening comprehension, which often allow for multiple possible answers.

One can ask at this point why one does not always perform all NLP analyses for every activity. Essentially the answer can be boiled down to 'don't guess what you know.' The more we know the linguistic properties, the types of variation, and the potential errors that the NLP needs to detect, the more specific information we can diagnose with higher reliability.

Naturally, the more activity types and NLP resources are present in a system, the more beneficial the kind of architecture presented here becomes – an issue we return to in section 4.2.

3.1.3 Feedback messages

Information about specific activity types also impacts the feedback messages that can be displayed to the student. Feedback messages for reading, listening and description activities should prioritize meaning over form. In the case of TAGARELA, if multiple errors are diagnosed, meaning-based errors will be displayed first for these types of activities. Feedback messages for rephrasing activities, on the other hand, can focus on syntactic errors at the sentence level. While vocabulary and fill-in-the-blanks activities tend to require feedback messages that target specific misuses of lexical items or morphemes.

3.2 Combining information from different NLP modules

A flexible, annotation-based ICALL architecture is also motivated by the need to support interleaving of contributions from different modules. For instance, tokenization can already resolve some part-of-speech ambiguities. Take, for example, the Portuguese token *a*, which can be a preposition (*to*), a pronoun (*her*, clitic direct object), or an article (*the*, feminine singular). But when the token *a* arises in the analysis of a contraction, such as in (8), the correct part-of-speech can unambiguously be determined.

(8) a. da = de + $a_{article}$

b. vê-la = ver + $a_{clitic\ pronoun}$

c. à = $a_{preposition}$ + $a_{article}$

The tokenizer thus should already be able to assign the part of speech in such cases. To support this, in an annotation-based architecture the same data structure is accessed by the NLP modules and can be enriched monotonically.

Another example for several NLP modules contributing to the same representation is part-of-speech disambiguation. For the lexical ambiguity which arises at the point of lexical lookup, disambiguation can be based on two distinct sources of information. On the one hand, the Constraint Grammar-like disambiguation rules introduced in section 2.1 attempt to use information about the local context to reduce or eliminate the ambiguity. On the other hand, the frequency of a given tag as specified in the CURUPIRA lexicon is used for disambiguation of any remaining cases, and one can readily imagine the integration of more complex statistical ambiguity resolution modules.

3.3 Combining information from different sources: Learner input, activity model, learner model

In addition to the information obtained from the learner input, information about the learner and the activity performed can also play an important role and thus needs to be integrated into the ILTS architecture.

3.3.1 Integrating information from the activity model

The issue essentially brings us back to the very beginning of the paper, where we stated that for some constrained exercise types it is possible to anticipate and hard-code for each potential input the corresponding feedback, whereas for others it is necessary to abstract and generalize using NLP techniques. Important for us here is that even for exercise types leading to a wider range of well-formed and ill-formed input, it is possible to hand-specify certain cases and the feedback to be provided for them. This insight can be useful to avoid costly NLP steps (either to avoid them completely, or to only run them once and then cache the result), or to manually provide information which cannot be reliably identified or identified at all using available NLP techniques. It thus is an important requirement for an ILTS processing architecture to support the flexible integration of the NLP analysis of the learner input with hand-encoded information for specific learner inputs provided as part of the activity models.

Interestingly, one can also see this flexible integration of static information from the activity model with the dynamic identification of information obtained by analyzing the learner input as presenting us with a continuum of systems stretching all the way from traditional CALL systems, where all learner input and the feedback to be provided for it needs to be anticipated and hard-coded, to an ICALL system without explicit activity information (such as a system providing feedback on free form essays), where all feedback is provided based exclusively on information derived by processing the learner input using general NLP resources.

3.3.2 Integrating information from the learner model

Information about the learner, their typical language use and errors, and the strategies they use to perform a particular activity can be crucial for disambiguating the learner input (cf. Amaral & Meurers, 2008). It thus is necessary to obtain an architecture which flexibly integrates information from the learner model and activity model with the NLP analysis of the learner input, both in terms of the processing itself and in terms of combining the output of the different sources of information into a single annotated representation of the learner input.

4 Realization of the architecture in UIMA

Having motivated the use of a demand-driven, annotation-based NLP framework for ILTS, in this section we connect the conceptual issues to a concrete NLP architecture. We introduce the Unstructured Information Management architecture (UIMA) and describe how we used its features to realize the envisaged ILTS architecture.

4.1 What UIMA offers

UIMA (Ferrucci & Lally, 2004) is a general framework for the automatic annotation of unstructured data, such as audio or text. In practice, UIMA is almost exclusively used for NLP applications. Central to UIMA's design is the idea of storing analysis results in a shared repository, the Common Analysis System (CAS, cf. Götz & Suhre, 2004). The CAS stores the text to be annotated separately from the annotations that pertain to specific parts of the text, similar to stand-off XML annotation in current linguistic corpus annotation schemes (cf., e.g., Ide et al., 2000). Annotations can be seen as enrichment layers that add information. This contrasts with traditional NLP architectures transforming the input from one representation to another, depending on what each module in such a pipeline expects. In UIMA, the so-called *Annotator*s provide information by adding new annotations to the CAS. Subsequent modules can then retrieve previously added annotations and add new ones as needed.

In connection with the central data repository, UIMA introduces the idea of a global *type system*, where the types of annotation to be stored are defined. Annotations are described in terms of typed feature structures, where feature values can be of any type. For example, a type *Token* might have a feature `tag` with value of type *String*, which stores the token's part-of-speech tag. An advantage of type systems is that they enable meaning-based access; type definitions are explicitly established for the whole application, so every Annotator knows which information can be found where and no data structure checking is necessary.

4.2 Using UIMA to implement an ILTS architecture

4.2.1 Type system

Our type system is organized around three main units to which all other information is attached. *Token*s are the smallest unit, to which the result of spell checking and lexical lookup for part-of-speech and morphological information is annotated.

*Phrase*s represent partial or complete parse trees, built of tokens and other phrases. For this purpose, they have a list of daughter nodes that represent their subtrees. For convenience, one can also store a reference to the parent node of a phrase. Figure 2 exemplifies the structure for the *Phrase* 'menina bonita' ('beautiful girl'). Phrases can be annotated with the result of daughter agreement, such as subject-verb agreement.

$$\begin{bmatrix} \textit{Phrase} & \\ \text{\textsc{begin}} & \mathit{0} \\ \text{\textsc{end}} & \mathit{13} \\ \text{\textsc{category}} & \textit{`NP'} \\ \text{\textsc{children}} & \left\langle \begin{bmatrix} \textit{Token} & \\ \text{\textsc{begin}} & \mathit{0} \\ \text{\textsc{end}} & \mathit{6} \\ \text{\textsc{category}} & \textit{`noun'} \\ \text{\textsc{Spelling}} & \begin{bmatrix}\textit{Spelling}\\ \dots\end{bmatrix} \\ \text{\textsc{lexDef}} & \left\langle \begin{bmatrix}\textit{LexiconInfo}\\ \dots\end{bmatrix}\right\rangle \\ \text{\textsc{tokenString}} & \textit{`menina'}\end{bmatrix}, \begin{bmatrix} \textit{Phrase} & \\ \text{\textsc{begin}} & \mathit{7} \\ \text{\textsc{end}} & \mathit{13} \\ \text{\textsc{category}} & \textit{`AP'} \\ \text{\textsc{children}} & \left\langle \begin{bmatrix}\textit{Token} & \\ \text{\textsc{category}} & \textit{`adj'} \\ \text{\textsc{tokenString}} & \textit{`bonita'} \\ \dots & \end{bmatrix}\right\rangle \\ \text{\textsc{parent}} & \begin{bmatrix}\textit{Phrase}\\ \dots\end{bmatrix}\end{bmatrix} \right\rangle \end{bmatrix}$$

Figure 2: Typed feature structure representation of the phrase *menina bonita*

The third type of annotation, *AnalysisResults*, does not refer to a particular portion of the input string but rather to the input as a whole. It is a repository of information which can be directly used by the diagnosis part of the system, such as the result of global agreement checking and the content assessment module introduced in section 2.1.

A notable aspect of the *Token* type is that it can associate an underlying form. For example, as we saw in section 3.2, certain Portuguese words such as contractions are syntactically complex, i.e., they can be thought of as consisting of two underlying tokens. The contraction 'da' we saw in (8a) is analyzed as consisting of the preposition 'de' (of) and the determiner 'a' (the). During the analysis, modules such as the parser need to refer to 'de' and 'a' separately if phrase structure rules are to apply properly. However, as discussed in Amaral & Meurers (2009), feedback to the learner needs to be given in terms of the surface representation, in this case 'da'. To address the need to encode both perspectives, a *Token* can store a list of underlying *Token*s under `deepForm` as illustrated in Figure 3.

$$\begin{bmatrix} \textit{Token} & \\ \textsc{begin} & \textit{0} \\ \textsc{end} & \textit{2} \\ \textsc{tokenString} & \textit{`da'} \\ \textsc{category} & \textit{`prep'} \\ \textsc{lexDef} & \left\langle \begin{bmatrix} \textit{LexiconInfo} & \\ \textsc{pos} & \textit{`prep'} \\ \textsc{canonic} & \textit{`da'} \\ \textsc{frequency} & \textit{35} \\ \textsc{source} & \textit{`lexicon'} \end{bmatrix} \right\rangle \\ \textsc{deepForm} & \left\langle \begin{bmatrix} \textit{Token} & \\ \textsc{begin} & \textit{0} \\ \textsc{end} & \textit{2} \\ \textsc{tokenString} & \textit{`de'} \\ \textsc{category} & \textit{`prep'} \\ \textsc{lexDef} & \left\langle \begin{bmatrix} \textit{LexiconInfo} & \\ \textsc{pos} & \textit{`prep'} \\ \textsc{source} & \textit{`tokn'} \end{bmatrix} \right\rangle \end{bmatrix}, \begin{bmatrix} \textit{Token} & \\ \textsc{begin} & \textit{0} \\ \textsc{end} & \textit{2} \\ \textsc{tokenString} & \textit{`a'} \\ \textsc{category} & \textit{`det'} \\ \textsc{lexDef} & \left\langle \begin{bmatrix} \textit{LexiconInfo} & \\ \textsc{pos} & \textit{`det'} \\ \textsc{source} & \textit{`tokn'} \\ \textsc{gender} & \textit{`f'} \\ \textsc{number} & \textit{`s'} \end{bmatrix} \right\rangle \end{bmatrix} \right\rangle \end{bmatrix}$$

Figure 3: Typed feature structure representation of the contraction *da* ('of the')

4.2.2 Multiple views for learner and target answers

Similar to other ILTS, TAGARELA uses target answers as a reference for analysis of learner answers. Both target and learner answers need to be processed (tokenized, part-of-speech tagged, etc.) independently, but certain analysis steps need to make reference to both of the annotated representations. For example, the content assessment modules need to map tokens in the learner answer to the ones in the target answer. Consequently, we need a data structure to encode such mappings. To avoid doubly encoded information, it should also provide access to the annotated analysis results for both learner and target inputs. UIMA provides an adequate solution called *multiple views*. In contrast to a regular CAS, a multi-view CAS can hold more than one text. This allows us to analyze learner and target answers independently and to create mappings using the representations of either part.

Consider, for example, the ill-formed learner input ‘Ele se chamo Carlos’ (‘His name is Carlos’), where the learner made a wordform error in the verb ‘chama’, using first-person ‘chamo’ instead. This can be encoded through a mapping from the learner token ‘chamo’ to the target token ‘chama’. The mapping also makes the token annotations, such as the morphological information accessible, allowing the system to pinpoint the exact nature of the mismatch when giving feedback to the learner.

4.2.3 Input and output specifications for analysis modules

We argued in section 3.1 that different activities require different NLP analysis. So how can the different activity demands flexibly be translated into processing strategies? Our solution to this problem makes use of the explicitness enforced by UIMA type systems. Given that types have to be declared before any analysis is done, they can be used as a means of specifying analysis requirements. For example, a fill-in-the-blank activity can require *Token*s with a specified `lexDef` feature, which means that lexical lookup must be done in addition to tokenization.

Binding activities to certain annotation types has the drawback that it only takes into account the NLP analysis – but the ultimate goal of an ILTS is to give useful feedback. We therefore added explicit error types to the system in order to fill in the missing link. The strategy can be described as top-down: pedagogical intervention opportunities are expressed as targeted error types, i.e., the particular errors a specific activity focuses on. These error types are mapped to required annotation types that are then annotated by the relevant analysis modules. The system thus knows what errors to focus on in processing and what to prioritize in the feedback for a given activity model. We elaborate on this process with examples in section 4.3.

In addition to formulating the overall analysis requirements, annotation types are also used to express dependencies between individual analysis modules. The lexicon lookup, for example, needs to work on tokenized input data. Hence, the input specification for the lexicon module is required to contain at least the type *Token*. The output specification then additionally contains the feature `lexDef`, whose value is a set of lexical entries.

4.3 Handling a range of activity types

We can now turn to describing in more detail, how the variation in TAGARELA's activities outlined in section 3.1 can be handled by our architecture. As mentioned in the previous section, our architecture dynamically adapts processing and feedback to the activity using the error types specified in the activity model.

The strategy employed by the first version of TAGARELA (Amaral, 2007) was to always prefer feedback on meaning over feedback on form. In our UIMA-based reimplementation of TAGARELA, the strategies are specified as part of each activity model to be able to adapt the strategy to the activity. We motivate the choice of different, activity-dependent strategies by describing two activity types, Reading Comprehension and Rephrasing, with sample learner input and generated feedback.

4.3.1 Reading comprehension

For a reading comprehension activity, the original strategy of preferring feedback on meaning over feedback on form makes good sense because the questions aim to test the learner's understanding of a given text. They are not designed to test certain grammatical constructions or morphological characteristics. Hence, the activity model for reading comprehension activities explicitly states that feedback should concentrate on any kind of meaning-based error that can be detected. Concretely, such errors can be inappropriate lexical choices or missing concepts, among others. Let us consider an example task from TAGARELA where the text describes a woman named Patrícia. One of the questions is 'Quantos anos ela tem?' ('How old is she?'). A possible learner answer is given in example (9).

(9) Ela **é** quinze **ano**.
she is fifteen year

There are two errors in this learner sentence. First, age here is expressed using the verb 'ser' ('to be') instead of the correct verb 'ter' ('to have') used for this purpose in Portuguese, so the learner made a lexical choice error. Second, 'ano' ('year') should be in its plural form 'anos' ('years') in order to agree with 'quinze' ('fifteen') in number. Both errors are detected by the system. However, since meaning-based feedback is preferred according to the activity model, TAGARELA selects the wrong lexical choice as the more important error and responds with the error message we already saw in Figure 1, saying "I am not expecting the verb 'ser' for this answer. Try using 'ter' instead." Once the learner has changed the verb and re-submitted his answer, the system reports the remaining agreement error.

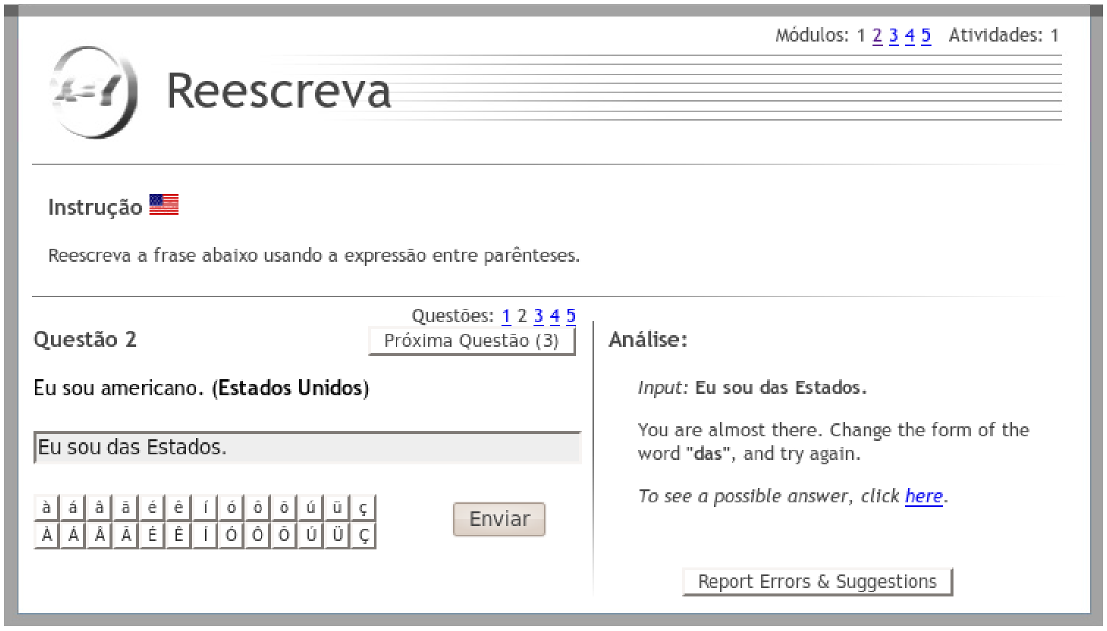
4.3.2 Rephrasing

TAGARELA's original 'meaning-over-form' strategy works fine for content-oriented activities such as answering reading comprehension questions, but for activities such as rephrasing the content is explicitly given. For those activities, the focus is on circumscribing a given sentence in a different way, using particular lexical material. Consequently, the activity model for rephrasing activities in the reimplemented TAGARELA system states that form errors should be the main target of feedback.

One such activity requires the learner to rephrase the sentence 'Eu sou americano' ('I am American') using the expression 'Estados Unidos' ('United States'). The intended correct target is 'Eu sou dos Estados Unidos' ('I am from the United States'). Consider the erroneous answer provided by a learner in (10).

(10)	Eu	sou	**das**	Estados.
	I	am	from the$_{fem}$	States$_{masc}$

There are two different errors. First, the learner forgot to include part of the proper name, the modifier 'Unidos' ('United'), in the rephrased sentence, which according to TAGARELA is a meaning-based error. Second, 'das' ('from the') has the wrong gender, yielding a form-based error. The particular activity model instructs TAGARELA to prioritize the form-based error, resulting in the message shown in Figure 4. Once this form error is fixed, TAGARELA provides feedback on the lexical content error.


Módulos: 1 2 3 4 5 Atividades: 1
Reescreva
Instrução
Questões: 1 2 3 4 5
Próxima Questão (3)
Análise:
à á â ã é ê í ó ô õ ú ü ç
À Á Â Ã É Ê Í Ó Ô Õ Ú Ü Ç
Enviar
Report Errors & Suggestions

Reescreva a frase abaixo usando a expressão entre parênteses.

Questão 2

Eu sou americano. (**Estados Unidos**)

Eu sou das Estados.

Input: **Eu sou das Estados.**

You are almost there. Change the form of the word "**das**", and try again.

To see a possible answer, click here.

Figure 4: Form-focused feedback on a Rephrasing activity

In summary, these two cases are meant to illustrate that due to the demand-driven architecture employed in the reimplemented TAGARELA, the activity designer is given some control over the behavior of the system in case of multiple learner errors. By encoding targeted error types into the activity model, the system can be told what feedback to prioritize. And as outlined in section 4.2.3, these error types are also used to guide the system's NLP.

5 Conclusion

In this paper, we discussed what we believe is needed to develop ILT systems capable of integrating a wider range of FLT activity types, resulting in learner input of a heterogeneous nature and evaluated on different criteria. We argued that in place of the fixed, uniform NLP pipelines employed by current ILTS, it is beneficial to view the NLP modules as enriching the input with annotations, giving the activity model control over which annotations can or must be provided in a demand-driven architecture. We showed how such an architecture can be realized in the UIMA architecture developed for general NLP analysis of unstructured information.

The UIMA-based reimplementation of TAGARELA (Ziai, 2009) offers the full functionality of the original system described in Amaral (2007). Additionally, it flexibly adjusts feedback both to the activity and to the learner, based on properties of the respective models. Using the popular UIMA framework, exchangeability of individual NLP components is made easier, which also makes porting the general architecture to other languages simpler. A system for Spanish is already being planned. The source code of the UIMA components will also be made available under an open-source license to encourage collaborative development with other researchers. This aspect of using UIMA as a standard architecture is in line with the recent argument of Wood (2008) for such standardization to facilitate compatibility and reusability of resources in ICALL development.

Making use of the increased modularity, we also plan to extend the system with the full student model proposed in Amaral & Meurers (2008). In the current version, task strategies such as scanning a text are not associated with activities yet and thus cannot be included in the student model. The new modular architecture should enable us to implement these ideas with little technical overhead.

Finally, let us mention a striking parallel between the issue of bridging between the complex FLT needs and the ILTS architecture discussed in this paper and the complex business demands handled in current Service Oriented Architectures (SOA). More specifically, the adaptivity and dynamic configuration of the NLP processing sequence informed by the activity model and driven by the feedback needs that we have argued for in this paper seems to bear an interesting resemblance to the Case Handling approach which has been proposed in the context of business process management, for which Weske et al. (2004, pp. 3f) argue: "While straight-through processing strives for more automation, case handling addresses the problem that many processes are much too variable or too complex to capture in a process diagram (van der Aalst & Berens, 2001). One way to do this is to make workflows data-driven rather than process-driven and allow for authorizations to skip or undo activities."

Acknowledgements

We would like to thank the organizers and audience of the Symposium on NLP in CALL at EUROCALL 2007, where the ideas worked out in this paper were first presented. We are also grateful to Mat Schulze for providing insightful and encouraging comments on an earlier version of this paper. Finally, we discovered the relation to case handling in Service Oriented Architectures thanks to a presentation by Christoph F. Strnadl and the helpful pointers he provided.

References

Amaral, L. (2007). Designing Intelligent Language Tutoring Systems: Integrating Natural Language Processing technology into foreign language teaching. Ph.D. thesis, The Ohio State University.

Amaral, L. & D. Meurers (2008). From Recording Linguistic Competence to Supporting Inferences about Language Acquisition in Context: Extending the Conceptualization of Student Models for Intelligent Computer-Assisted Language Learning. *Computer-Assisted Language Learning* 21(4), 323–338. URL `http://purl.org/dm/papers/amaral-meurers-call08.html`.

Amaral, L. & D. Meurers (2009). Little Things With Big Effects: On the Identification and Interpretation of Tokens for Error Diagnosis in ICALL. *CALICO Journal* 27(1). URL `http://purl.org/dm/papers/amaral-meurers-09.html`.

Amaral, L. & D. Meurers (submitted). On Using Intelligent Computer-Assisted Language Learning in Real-Life Foreign Language Teaching and Learning URL `http://purl.org/dm/papers/amaral-meurers-10.html`.

Bailey, S. & D. Meurers (2008). Diagnosing meaning errors in short answers to reading comprehension questions. In *Proceedings of the 3rd Workshop on Innovative Use of NLP for Building Educational Applications, at ACL*. pp. 107–115. URL `http://aclweb.org/anthology-new/W/W08/W08-0913.pdf`.

Bick, E. (2000). *The Parsing System "Palavras": Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework*. Aarhus Univ. Press.

Bick, E. (2004). PaNoLa: Integrating Constraint Grammar and CALL. In H. Holmboe (ed.), *Nordic Language Technology (Yearbook 2003)*, Copenhagen: Museum Tusculanum, pp. 183–190.

Delmonte, R. (2003). Linguistic knowledge and reasoning for error diagnosis and feedback generation. *CALICO Journal* 20(3), 513–532. URL `http://purl.org/calico/delmonte-03.html`.

Ferrucci, D. & A. Lally (2004). UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering* 10(3–4), 327–348.

Götz, T. & O. Suhre (2004). Design and implementation of the UIMA Common Analysis System. *IBM Systems Journal* 43(3), 476–489.

Heift, T. (2003). Multiple Learner Errors and Meaningful Feedback: A Challenge for ICALL Systems. *CALICO Journal* 20(3), 533–548. URL `http://purl.org/calico/heift-03.html`.

Heift, T. & M. Schulze (2007). *Errors and Intelligence in Computer-Assisted Language Learning: Parsers and Pedagogues*. Routledge.

Holland, V., J. Kaplan & M. Sams (eds.) (1995). *Intelligent Language Tutors. Theory Shaping Technology*. New Jersey: Lawrence Erlbaum Associates, Inc.

Ide, N., P. Bonhomme & L. Romary (2000). XCES: An XML-based Encoding Standard for Linguistic Corpora. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*. pp. 825–830. URL `http://www.cs.vassar.edu/~ide/papers`.

Karlsson, F., A. Voutilainen, J. Heikkilä & A. Anttila (eds.) (1995). *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Berlin and New York: Mouton de Gruyter.

Levin, L. S. & D. A. Evans (1995). ALICE-chan: A case study in ICALL theory and practice. In Holland et al. (1995), pp. 77–98.

Martins, R., R. Hasegawa & M. das Graças Nunes (2006). Curupira: a functional parser for Brazilian Portuguese. In *Computational Processing of the Portuguese Language, 6th Int. Workshop, PROPOR. LNCS 2721*. Springer. URL `http://www.springerlink.com/content/b48vjft1l88yvrj0/fulltext.pdf`.

Nagata, N. (2002). BANZAI: An Application of Natural Language Processingto Web based Language Learning. *CALICO Journal* 19(3), 583–599. URL `http://purl.org/calico/nagata-02.html`.

Nagata, N. (2009). Robo-Sensei's NLP-Based Error Detection and Feedback Generation. *CALICO Journal* 26(3), 562–579.

Rypa, M. & K. Feuerman (1995). CALLE: An exploratory environment for foreign language learning. In Holland et al. (1995), pp. 55–76.

van der Aalst, W. M. P. & P. J. S. Berens (2001). Beyond workflow management: product-driven case handling. In S. Ellis, T. Rodden & I. Zigurs (eds.), *International ACM SIGGROUP Conference on Supporting Group Work (GROUP 2001)*. New York: ACM Press, pp. 42—-51.

Weske, M., W. M. P. van der Aalst & H. M. W. Verbeek (2004). Advances in Business Process Management. *Data & Knowledge Engineering* 50, 1–8.

Wood, P. (2008). Developing ICALL Tools Using GATE. *Computer Assisted Language Learning* 21(4), 383–392.

Ziai, R. (2009). A Flexible Annotation-Based Architecture for Intelligent Language Tutoring Systems. Master's thesis, Universität Tübingen.