Adaptive Corpus-Based Language Learning System Using Morphological and Dependency Analysis

Zuhair Mehdee

Memorial University of Newfoundland

CS 4750: Natural Language Processing

Todd Wareham

December 5, 2025

Abstract

This project presents the design and implementation of an adaptive language learning system targeted at intermediate students of Spanish. Unlike traditional flashcard applications that rely on static string matching and atomic vocabulary tracking, this system utilizes Natural Language Processing (NLP) techniques, specifically dependency parsing and morphological analysis, to provide granular feedback and adaptive content scheduling. The system parses student input into dependency trees to evaluate correctness independent of Spanish's flexible word order. Furthermore, it tracks learner proficiency not by sentence, but by decomposing content into specific skills (lemmas and morphological features). By integrating these micro-skills with the Free Spaced Repetition Scheduler (FSRS), the system dynamically selects corpus sentences that target the learner's specific grammatical and lexical weaknesses, addressing the "data sparsity" problem often found in Computer-Assisted Language Learning (CALL).

---

1. Introduction

Second Language Acquisition (SLA) requires exposure to comprehensible input that is slightly above the learner's current proficiency level. While beginner learners benefit from rote memorization of vocabulary, intermediate learners face a "plateau" where they must master complex syntactic structures and morphological variations.

Traditional Computer-Assisted Language Learning (CALL) tools often fail to support this stage effectively due to two primary limitations. First, they typically rely on linear string comparison for error checking. If a student produces a sentence that is grammatically correct but uses a different word order than the reference translation—a common occurrence in languages like

Spanish—the system marks it as incorrect. This discourages the learner and reinforces a rigid, unnatural usage of the target language.

Second, most systems track progress at the level of the "card" or the "sentence." As noted in foundational research by Duolingo (Settles & Meeder, 2016), this leads to a data sparsity problem. A student may successfully conjugate the verb *comer* (to eat) in the present tense but fail in the subjunctive. If the system treats "comes" and "comas" as entirely separate entities, it fails to model the student's actual knowledge gap: the *concept* of the subjunctive mood.

This project addresses these issues by developing a system that combines:

1. **Transformer-based Dependency Parsing:** To evaluate student input based on syntactic structure rather than linear sequence.
2. **Granular Skill Tracking:** To monitor proficiency at the level of lemma (vocabulary) and morphology (grammar) independently.
3. **Adaptive Corpus Selection:** To algorithmically select practice sentences that maximize the density of "urgent" skills for the learner.

---

2. Related Work

The intersection of NLP and language learning has been a fertile ground for research, particularly regarding Spaced Repetition Systems (SRS) and Automated Grammatical Error Detection.

2.1 Spaced Repetition and the Sparsity Problem

Spaced repetition, based on Ebbinghaus's forgetting curve, is the standard for vocabulary retention. Algorithms like SuperMemo (Wozniak, 1990) and Anki optimize intervals to maximize retention. However, applying SRS to language *generation* (sentences) is complex.

In their influential paper "A Trainable Spaced Repetition Model for Language Learning," Settles and Meeder (2016) analyzed data from Duolingo and highlighted a critical issue: the choice of features for the student model. They observed that tracking every unique word form (lexeme) results in data sparsity; a student sees the word *run* frequently, but *ran* rarely, making it difficult for the model to predict retention for the latter. They proposed Half-Life Regression (HLR) and suggested using "lexical features" (like morphological tags) to generalize knowledge. My project directly implements this suggestion by decoupling the *lemma* (the root meaning) from its *morphological tags* (Tense, Mood, Person). By tracking these as separate variables in the FSRS algorithm, the system allows success in one conjugation to inform the stability of the lemma, while failures in specific tenses isolate the grammatical deficit.

2.2 Dependency Parsing in CALL

Standard error correction often uses Levenshtein distance (edit distance), which is insufficient for languages with flexible syntax. Research by Chodorow et al. (2012) demonstrated that dependency trees—which map grammatical relations between words—provide a robust framework for error detection. By checking if the *subject* and *verb* agree, regardless of their position in the sentence, dependency parsing offers a pedagogical advantage over n-gram matching. This project utilizes the es_dep_news_trf transformer pipeline from spaCy (Honnibal et al., 2020) to implement such a parser-based evaluation.

3. System Architecture

The system is composed of three primary modules: the Corpus Processor, the Recursive Dependency Matcher (Grader), and the Adaptive Scheduler.

3.1 Corpus Preprocessing and Skill Extraction

The system utilizes a corpus of Spanish sentences. Each sentence is passed through the spaCy transformer pipeline to generate a dependency tree. The system then extracts a set of "skills" for every token in the sentence.

For example, the token *comas* (you eat [subjunctive]) is decomposed into:

- **Lemma Skill:** lemma:comer (captures vocabulary knowledge).
- **Morphological Skills:** morph:Mood=Subj, morph:Tense=Pres, morph:Person=2.

This decomposition allows the system to credit the user for knowing the vocabulary word *comer* even if they conjugate it incorrectly, correcting the "all-or-nothing" grading flaw of traditional apps.

3.2 Recursive Dependency-Based Comparison

To handle Spanish word order flexibility (e.g., *Yo como la manzana* vs. *La manzana la como yo*), the system employs a recursive tree-traversal algorithm rather than string matching.

The Algorithm:

1. **Root Identification:** The system identifies the ROOT verb in both the reference and student sentences.

2. **Recursive Matching:** It compares the root lemmas. If they match, it recursively retrieves the children of the root (Subject, Object, Modifiers).

3. **Cross-Positional Alignment:** Children are matched by their dependency labels (e.g., nsubj, obj), not their linear index. This allows the subject to appear either before or after the verb without triggering an error.

4. **Rescue Logic:** A specific innovation in this project is "Rescue Logic." If a student substitutes the root verb (e.g., using *decir* instead of *hablar*), the dependency labels of the children may shift (an *indirect object* might become a *direct object*). The system detects this mismatch and searches the pool of unmatched words for lemmas that exist in the student input, "rescuing" them from being marked as errors. This ensures that a single vocabulary substitution does not cascade into a false "grammatical structure" failure.

3.3 Adaptive Sentence Selection via FSRS

The system integrates the Python implementation of the Free Spaced Repetition Scheduler (FSRS v5). The FSRS model calculates the "retrievability" and "stability" of every skill in the database.

When the user requests a new practice session, the system scans the corpus and scores every available sentence using a **normalized urgency formula**:

$$Score(S) = \frac{\sum_{k \in S} Urgency(k)}{\sqrt{|S|}}$$

Where ***Urgency(k)*** is derived from the days overdue and difficulty of skill ***k***. The square root normalization prevents the system from biasing purely toward long sentences, favoring instead sentences that are *dense* with high-priority (weak) skills.

---

4. Implementation Details

The system was implemented in Python. Key libraries included spacy for NLP tasks and fsrs for the spaced repetition logic.

4.1 Technology Stack

- **Language:** Python 3.10
- **NLP Engine:** spaCy with es_dep_news_trf (Transformer-based Spanish model). The transformer model was chosen over the lighter statistical models (es_core_news_sm) because accurate dependency labeling is critical for the recursive grading logic.
- **Scheduler:** fsrs (Python bindings for the Rust FSRS implementation).
- **Data Structure:** An in-memory dictionary simulates the database, mapping Skill IDs (strings) to FSRS Card objects.

4.2 Handling Morphological Ambiguity

One challenge encountered was the handling of null subjects (implicit pronouns), which are mandatory in English but optional in Spanish. The Recursive Matcher was adjusted to ignore "Omission" errors for pronouns (e.g., *yo*, *tú*) if the verb morphology correctly encoded the person and number. This required checking the morph dictionary of the verb node when a child node was missing.

5. Results and Discussion

5.1 Robustness to Word Order

Qualitative testing demonstrated the system's ability to correctly grade syntactically valid variations.

- *Reference:* "El hombre le dio el libro a ella." (SVO)
- Student: "A ella le dio el libro el hombre." (OVS)

  Result: The recursive matcher successfully aligned "hombre" (nsubj), "libro" (obj), and "ella" (iobj) despite the complete inversion of linear order. A standard Levenshtein distance algorithm would have marked this translation as nearly 100% incorrect.

5.2 Efficacy of Skill Decomposition

The separation of lemma and morphology proved effective in simulation. When the simulated user repeatedly failed sentences containing Subjunctive verbs, the FSRS algorithm increased the urgency of the morph:Mood=Subj skill. Consequently, the adaptive selector began presenting sentences containing different verbs but utilizing the Subjunctive mood. This achieves the goal of "concept-based" learning rather than isolated rote memorization.

5.3 Limitations

The system currently relies on the accuracy of the spaCy parser. If the student produces a sentence with complex errors that confuses the parser, the dependency tree may become malformed, leading to confusing feedback. Additionally, the system currently lacks a semantic embedding layer; thus, valid synonyms (e.g., *coche* vs. *auto*) are marked as vocabulary errors unless explicitly hardcoded in the rescue logic.

Conclusion

This project demonstrates a viable architecture for an intermediate-level language learning system that transcends the limitations of traditional flashcards. By applying dependency parsing, the system respects the syntactic flexibility of the target language. By implementing the granular feature tracking proposed by Settles and Meeder (2016), it solves the data sparsity problem, allowing the system to track grammatical concepts across different vocabulary items. Future work will focus on integrating semantic vector spaces (Word2Vec or BERT) to handle synonymy and scaling the corpus to include paragraph-level contexts.

References

Chodorow, M., Tetreault, J., & Han, N. (2012). Detection of Grammatical Errors Involving Prepositions. *Proceedings of the 4th Workshop on Innovative Use of NLP for Building Educational Applications*, 25-30. Association for Computational Linguistics.

Honnibal, M., Montani, I., Van Landeghem, S., & Boyd, A. (2020). spaCy: Industrial-strength Natural Language Processing in Python. *Zenodo*. https://doi.org/10.5281/zenodo.1212303

Krashen, S. D. (1982). *Principles and Practice in Second Language Acquisition*. Pergamon Press.

Settles, B., & Meeder, B. (2016). A Trainable Spaced Repetition Model for Language Learning. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), 1284–1293. Berlin, Germany. Association for Computational Linguistics. https://research.duolingo.com/papers/settles.acl16.pdf

Wozniak, P. A. (1990). Optimization of learning. *Master's Thesis, University of Technology in Poznan*.

Ye, J., & Jiang, J. (2023). FSRS: Free Spaced Repetition Scheduler. *GitHub Repository*. https://github.com/open-spaced-repetition/fsrs