# Automatic Recognition of Chinese Separable Verbs (Liheci)

## 1. Introduction

The accurate identification of Chinese separable verbs (离合词, liheci) remains a significant challenge for current POS taggers and word segmentation systems. Existing tools like HanLP often fail to recognize the complex structures of liheci, leading to errors in both automatic processing and language learning applications.

This is particularly problematic for non-native learners, who struggle with questions like: Should "的" be inserted after a pronoun? Can a prepositional phrase serve as a valid insertion? These subtle grammatical rules are rarely handled well by existing systems.

This project develops a robust, four-stage pipeline for automatic liheci recognition, achieving **91.88% precision** and **98.66% recall** (F1: 95.16%) on 284 test instances. The system formalizes detailed recognition rules that provide both computational accuracy and educational value.

## 2. Project Structure and Files

### 2.1 Data Files

| File | Description |
|---|---|
| `data/liheci_lexicon.csv` | **131 liheci entries** with linguistic features: head (A), tail (B), type, pinyin, RedupPattern (55 marked AAB), PronounInsertion rules (REQUIRE_DE/PREFER_DE/OBJ_OK/NO_DIRECT_NP), PPRequirement |
| `data/test_sentences.txt` | **284 test instances**: 149 True cases + 135 False cases (7 error types) |

### 2.2 Pipeline Scripts

| Stage | Script | Function |
|---|---|---|
| Generation | `01.generate_liheci_split_xfst.py` | Generates XFST rules for WHOLE/SPLIT recognition |
| Generation | `02.generate_liheci_redup_xfst.py` | Generates XFST rules for AAB reduplication |
| Stage 1 | `03.stage1_split_whole_recognition.py` | Runs FST recognition for 131 lemmas |
| Stage 2 | `04.stage2_redup_recognition.py` | Validates reduplication for 55 AAB lemmas |
| Generation | `05.generate_insertion_context_xfst.py` | Generates character-level annotator FST |
| Stage 3 | `06.stage3_insertion_analysis.py` | Insertion analysis + confidence filtering + linguistic rules |
| Stage 4 | `07.stage4_pos_validation.py` | HanLP POS validation for head/tail |

| Stage | Script | Function |
|---|---|---|
| Evaluation | `evaluate_by06.py`, `evaluate_by07.py` | Performance metrics computation |

### 2.3 Output Files

- `outputs/liheci_hfst_outputs.tsv` — Stage 2 output (190 rows)
- `outputs/liheci_insertion_analysis.tsv` — Stage 3 output (170 rows)
- `outputs/liheci_pos_validated.tsv` — Stage 4 final output (160 rows)
- `outputs/liheci_pos_rejected.tsv` — Stage 4 rejected (10 rows)

## 3. Pipeline Design and Key Decisions

### 3.1 Data Preparation: Lexicon Features

The lexicon (`liheci_lexicon.csv`) contains rich linguistic annotations exploited in later stages:

- **RedupPattern='AAB'**: 55/131 lemmas allow reduplication (散步→散散步). Used in Stage 2 to filter invalid reduplications like *结结婚.
- **PronounInsertion**: Four categories (REQUIRE_DE, PREFER_DE, OBJ_OK, NO_DIRECT_NP) encoding pronoun+的 constraints. Used in Stage 3.
- **PPRequirement='EXT_ONLY'**: Some liheci require external PP placement. Used in Stage 3.

### 3.2 Why Separate FSTs for Standard vs. Reduplication?

**Problem**: I initially attempted to combine WHOLE, SPLIT, and AAB reduplication patterns into a single FST. This caused **state explosion**—each of the 55 AAB-allowed lemmas requires multiple reduplication patterns (AAB, A一AB, A了AB), and combining these with the SPLIT patterns (which allow arbitrary insertions between head and tail) created too many state combinations, making compilation fail or become computationally intractable.

**Solution**: Two-stage FST approach:

- **Stage 1 FST**: Recognizes head...tail patterns for all 131 lemmas, regardless of what appears between them
- **Stage 2 FST**: Separately validates AAB/A一AB/A了AB patterns only for the 55 lemmas marked with `RedupPattern='AAB'`

**Result**: Stage 2 identified 132 potential reduplication patterns from 301 initial matches. After validation against the AAB-allowed list:

- 21 valid reduplications kept
- 90 invalid attempts filtered (e.g., *结结婚, *离离婚)
- **100% filtering rate** for INVALID_REDUP error type

### 3.3 Stage 3: Insertion Analysis Architecture

**Division of Labor**:

- `05.generate_insertion_context_xfst.py`: Generates FST that **only annotates** characters with grammatical tags (ASPECT, NUM, CLF, MOD, PRO, DE, RES, TIME). No decision-making.

- `06.stage3_insertion_analysis.py`: **Makes all decisions**—classifies insertion types, calculates confidence, applies linguistic rules, filters candidates.

**Confidence Score Calculation**:

```
confidence = tagged_characters / total_insertion_characters
```

- "了一个好" → 4/4 tagged = 1.0
- "热水澡" → 1/3 tagged (only 热:MOD) = 0.33

**Threshold Selection (0.3)**: Empirically chosen by examining the confidence distribution. This threshold effectively filters low-confidence coincidental matches (e.g., "桌**上**放着**课**本" matching 上课) while preserving true positives.

### 3.4 Stage 3: Linguistic Rules with Automatic Rejection

Beyond coverage-based confidence, certain linguistic violations trigger **automatic confidence = 0.0**:

These rules use lexicon's PronounInsertion field, and are implemented in liheci-analyzer\scripts\06.stage3_insertion_analysis.py

**Rule 1: Pronoun + DE Constraints**

| Lexicon Value | Rule | Example |
|---|---|---|
| REQUIRE_DE | Must have 的 after pronoun | 捣他**的**乱 ✓ / *捣他乱 → conf=0 |
| PREFER_DE | 的 preferred (penalty if missing) | 帮他**的**忙 ✓ / 帮他忙 → conf×0.7 |
| OBJ_OK | Pronoun as object allowed | 帮了**我**一个大忙 ✓ |
| NO_DIRECT_NP | No direct pronoun; use external PP | *见他面 → conf=0; **跟他**见面 ✓ |

**Rule 2: PP Position** (using lexicon's PPRequirement field)

- Some liheci require PP **before** head (external), not inside insertion
- Violation: PP inside insertion → confidence = 0.0
- Example: **跟他**见了个面 ✓ vs. *见**跟他**面 ✗

These rules filter **100% of MISSING_DE errors** (4/4) in the test set.

### 3.5 Why POS Validation Only in Stage 4 (Not Stage 3)?

**Problem**: HanLP's POS tagset (VV, NN, AD, etc.) is **too coarse** for insertion analysis. For example:

- "了一个好觉" needs fine-grained tags: 了:ASPECT, 一:NUM, 个:CLF, 好:MOD
- HanLP would tag these as AS, CD—losing the linguistic distinctions needed for confidence scoring

**Solution**: Use custom 8-tag FST for insertion analysis (Stage 3), then apply HanLP POS validation **only for head/tail** in Stage 4.

**Stage 4 POS Rules** (type-specific):

| Type | HEAD Allowed POS | TAIL Allowed POS | Notes |
|---|---|---|---|
| **Verb-Object** | VV only | NN/NR/NT, VV, M, VA | TAIL=VV for WHOLE/REDUP forms |
| **Pseudo V-O** | VV only | NN/NR/NT, VV | Similar to V-O |
| **Modifier-Head** | VV, VA, AD | NN/NR/NT, VA, VV | HEAD can be adjective/adverb |
| **SimplexWord** | VV only | NN/NR/NT, VV | Idiomatic compounds |

**TAIL Blacklist** (applies to all types):

- Functional words **cannot** be TAIL: AD (adverb), P (preposition), CS/CC (conjunction), DT (determiner), DEG/DEC (的/得), AS (aspect marker), SP (sentence particle)
- Rationale: If TAIL is tagged as a functional word, the match is likely a character coincidence, not a genuine liheci

**Example filtered**: "桌**上**放着**课**本" — "上" tagged as LC (localizer), not VV → HEAD invalid → rejected

# 4. Evaluation

## 4.1 Metrics

- **Precision** = TP / (TP + FP)
- **Recall** = TP / (TP + FN)
- **F1** = 2 × Precision × Recall / (Precision + Recall)

## 4.2 Stage-by-Stage Performance Comparison

| Stage | Output | Precision | Recall | F1 | TP | FP | FN |
|---|---|---|---|---|---|---|---|
| Stage 1+2 (FST) | 190 | 78.42% | 100.00% | 87.91% | 149 | 41 | 0 |
| Stage 3 (Insertion) | 167 | 87.43% | 97.99% | 92.41% | 146 | 21 | 3 |
| **Stage 4 (POS)** | **157** | **92.36%** | **97.32%** | **94.77%** | **145** | **12** | **4** |

**Key Observations**:

- **Stage 1+2**: Perfect recall (100%) but low precision (78.42%). FST captures all 149 true cases but includes 41 false positives.
- **Stage 3**: Precision +9.01pp (78.42%→87.43%) by filtering 20 FPs. Lost 3 TPs (FN: 0→3).
- **Stage 4**: Precision +4.93pp (87.43%→92.36%) by filtering 9 FPs via POS validation. Lost 1 additional TP (FN: 3→4).

## 4.3 Error Type Filtering Effectiveness

| Error Type | Total | Stage 2 FP | Final FP | Filtering Rate |
|---|---|---|---|---|
| INVALID_REDUP | 43 | 0 | 0 | **100%** |
| CROSS_CONST | 45 | 4 | ~1 | **97.8%** |

| Error Type | Total | Stage 2 FP | Final FP | Filtering Rate |
|---|---|---|---|---|
| MISSING_DE | 4 | 4 | 0 | **100%** |
| REVERSED_ORDER | 6 | 0 | 0 | **100%** |
| PP_POS | 10 | 10 | ~2 | **80%** |
| COINCIDENCE | 24 | 23 | ~10 | **58%** |

**Remaining Challenge**: 13 final FPs are mostly COINCIDENCE type—characters with valid POS patterns requiring semantic understanding (e.g., "贪**小**便宜" vs. "小便").

# 5. Conclusion

## 5.1 Summary

The four-stage pipeline achieves strong performance (Precision 92.36%, Recall 97.32%, F1 94.77%) by combining:

1. **FST pattern matching** for structural recognition
2. **Lexicon-driven linguistic rules** for grammatical constraints
3. **POS validation** for syntactic filtering

## 5.2 Limitations

**Unidirectional Pipeline**: The current system only supports **analysis** (sentence → liheci recognition), not **generation** (liheci + insertion pattern → sentence). A bidirectional system could generate example sentences like "帮 + PRO + 的 + 忙 → 帮他的忙", which would be highly valuable for Chinese language learners. Due to time constraints, this generation capability was not implemented.

**COINCIDENCE Errors**: The remaining 12 false positives are predominantly COINCIDENCE type—cases where characters happen to match a liheci pattern but are actually unrelated words. For example, "小便" (urinate) contains the characters "小" and "便", which coincidentally match the head and tail of "小便" as a potential liheci.

This is particularly challenging for **WHOLE forms** (unseparated liheci), where two characters appear adjacent without any insertion to analyze. Unlike SPLIT forms where insertion analysis provides confidence signals (e.g., "了一个好" → high confidence), WHOLE forms offer no such internal evidence. Distinguishing "他在**吹牛**" (genuine liheci: bragging) from "贪**小便**宜" (coincidence: "小便" is not functioning as a liheci) requires semantic understanding beyond surface patterns—a challenge not yet resolved in this system.

## 5.3 Insights

While no single hard rule applies universally to all 131 liheci, the combination of lexicon-encoded constraints and pattern-based confidence scoring captures the regularities that do exist. This formalization provides both computational utility and linguistic insight—potentially valuable for Chinese language education where learners struggle with questions like "Why is 帮他忙 acceptable but *见他面 is not?"

---

*Repository: https://github.com/Erin-lab-design/liheci-analyzer*