Text Normalization Challenge Report

Executive Summary

FST-based text normalization system for English cardinal numbers (0-1000) using Pynini. Compiles grammar to FAR file for efficient normalization, with Python fallback for portability.

1. Introduction

This Text normalization converts numeric strings to written-out forms (e.g., "21" = "twenty-one", "100" = "one hundred"). Critical for text-to-speech and speech recognition applications.

2. Methodology

2.1 Approach

Two-tier system:

1. **Primary**: FST-based using Pynini (`src/grammar.pynini -> src/grammar.far`)
2. **Fallback**: Pure Python rule-based normalizer in `src/normalize.py` in case

2.2 Grammar Design

Hierarchical FST structure:

Basic Components:

- Units (0-9): Direct digit-to-word mapping
- Teens (10-19): Special cases (ten, eleven, twelve, etc.)
- Tens (20-90): Multiples of ten (twenty, thirty, etc.)

Composite Numbers:

- Two-digit (20-99): Hyphenated forms (twenty-one, forty-five)
- Hundreds (100-999): Pattern `[hundreds_word] + " hundred" + [remainder]`
  - Examples: "100" = "one hundred", "101" = "one hundred one", "245" = "two hundred forty-five"

- Special case: "1000" = "one thousand"

FST Construction:

- Uses `pynini.cross()` for mappings, `pynini.union()` for combination, `.optimize()` for efficiency
- Final FST: `two_digit | hundreds | thousand`

2.3 Implementation

Number Detection:

- Regex pattern: `\b\d+\b` (word boundaries)
- Only 0-1000 normalized; others unchanged
- Leading zeros preserved

Normalization Process:

1. Scan text for numeric patterns
2. Check range [0, 1000]
3. Convert via FST lookup (if FAR available) or Python function
4. Preserve text structure (spacing, punctuation)

3. How to Run

Clone the repository

```
git clone https://github.com/AlexKalll/fst-number-normalizer.git
cd fst-number-normalizer
```

Create a Virtual Environment

```
python3 -m venv venv
source venv/bin/activate # or venv/Scripts/activate for Windows
```

Install Packages

```
pip install -r requirements.txt
```

Compile Grammar

```
python scripts/compile_grammar.py
```

Loads `src/grammar.pynini` , builds FST, writes to `src/grammar.far` .

Usage

```
python src/normalize.py "I have 21 cats and 3 dogs."
```

```
from src.normalize import normalize_text
result = normalize_text("She is 45 years old.")
```

Tests

```
pytest tests -v
```

4. Performance Metrics

- **Compilation time**: < 1 second
- **FAR file size**: ~50-150 KB in my case.
- Runtime (Python fallback even it's slow)

WER Optimization:

- Comprehensive coverage (all 0-1000 explicitly handled)
- Correct hyphenation (21-99)
- Proper spacing for hundreds

5. Grammar Design Rationale

Why FSTs?

- Deterministic (same input - same output)
- Efficient lookup and composition
- Composable with other FSTs

- Standard in NLP/speech processing

Design Choices:

1. Explicit enumeration for 100-999 ensures correctness and debuggability
2. Hierarchical structure (units - teens - tens - hundreds) for maintainability
3. Correct hyphenation usage
4. Python fallback for reproducibility

6. Testing

Comprehensive unit tests cover: single digits, teens, two-digit, hundreds, 1000, edge cases, multiple numbers, and text without numbers.

7. File Structure

```
fst-number-normalizer/
├── src/
│   ├── normalize.py          # Main normalizer (FST + fallback)
│   ├── grammar.pynini         # FST grammar source
│   └── grammar.far            # Compiled FAR file
├── scripts/
│   └── compile_grammar.py     # Grammar compilation
├── tests/
│   └── test_basic.py          # Unit tests
├── report/
│   └── report.md              # This report
├── requirements.txt
└── README.md
```

8. Conclusion

This project implements FST-based normalization for English cardinal numbers 0-1000. System is accurate, efficient, robust (with fallback), and well-documented. Grammar design prioritizes correctness through explicit enumeration.

9. References

- Pynini documentation: https://www.openfst.org/twiki/bin/view/FST/WebHome
- https://huggingface.co/datasets/DigitalUmuganda/Text_Normalization_Challenge_Unittests_Eng_Fra/blob/main/tutorial_FST_text_normalization.ipynb