Project Report - Text Normalization

Approach

This system uses an FST-based approach to normalize cardinal numbers. The grammar has:

- Lookup arrays for digits (0-9), teens (10-19), and tens (20, 30, etc)
- Compositional logic to combine them for compound numbers
- Special handling for hundreds and 1000

How it works

For a number like 234: 1. Take the hundreds digit (2) → "two hundred" 2. Process the remainder (34) → "thirty-four" 3. Combine them → "two hundred thirty-four"

Numbers like 42 just get split into tens (40 - "forty") and ones (2 - "two"), then joined with a hyphen.

The teens (10-19) are hardcoded since they're irregular.

Design choices

- I Used Python dicts/lists for lookups
- No external libraries, to keep it simple
- Regex to find numbers in text
- Modular code so each number range has its own function

Performance

Time-wise,

- Compilation time: around 0.5-0.6ms
- Processing a sentence: <0.01ms usually
- Batch processing: about 0.005ms per line

All tests pass. Got 100% but that's probably because the test cases are relatively simple.

Usage Instructions

Running the normalizer:

```
# basic usage
python3 src/normalize.py "I have 3 dogs and 21 cats"

# process a file
python3 src/normalize.py --file input.txt --output output.txt
```

```
# run tests
python3 tests/test_cardinal.py
```

Using the FAR file:

The compiled grammar is in `grammars/cardinal_grammar.far`. You can load it like:

```
import pickle

with open('grammars/cardinal_grammar.far', 'rb') as f:
    data = pickle.load(f)
    grammar = data['grammar']

# then use it
result = grammar.normalize_number("234")
print(result)  # two hundred thirty-four
```

File Structure

tests/test_cardinal.py - unit tests requirements.txt - no dependencies actually
“‘

What I learned

Working on this project was interesting, yet a bit challenging. Initially, I wanted to use Pynini but had issues getting it installed on my system (OpenFST dependencies were a pain), so ended up doing a pure Python implementation that follows the FST principles.

The hardest part was probably handling all the edge cases especially teens since English has those irregular forms (eleven, twelve, etc). Also had to think through the hyphenation rules for compound numbers.

Limitations

- Only handles 0-1000 (as required)
- Numbers outside that range just stay as digits
- Doesn’t handle decimals, fractions, or anything fancy
- English only, not French