SMT-BLEU Assignment: Statistical Machine Translation with BLEU Evaluation

Language Pair: English → Hindi
Primary SMT Toolkit: Moses
Fallback: Toy SMT (Python-based)

Table of Contents

Overview

This project implements a complete Statistical Machine Translation system for English→Hindi translation with comprehensive BLEU evaluation. The system includes:

- **Three Translation Systems:**
  1. **Moses SMT**: Industry-standard phrase-based SMT decoder
  2. **Toy SMT**: Custom implementation with phrase table + trigram LM + beam search
  3. **Word-by-Word**: Dictionary-based baseline
- **BLEU Implementation:**
  - From-scratch implementation (no black-box libraries)
  - Modified n-gram precision with clipping
  - Brevity penalty computation
  - Support for multiple references
  - Individual BLEU-1, BLEU-2, BLEU-3, BLEU-4, and cumulative BLEU
- **Web Interface:**
  - Streamlit-based interactive UI
  - Source text input and reference selection/upload
  - Comparative evaluation of multiple translation systems
  - Detailed n-gram precision tables and statistics

Project Structure

```
smt-bleu-assignment/
  README.md                    # This file
  Report.md                    # Technical report with architecture
  TaskB.md                     # BLEU improvement strategies (PDF-ready)
  LiteratureReview.md          # Literature survey on MT evaluation
  references.bib               # BibTeX references
  requirements.txt             # Python dependencies
  run_checks.sh                # Test and validation script
  SCREENSHOTS.md               # Screenshot checklist

  data/                        # Data files
     built_in_corpus.json     # 10+ English-Hindi parallel sentences
     phrase_table.json        # Phrase translations for Toy SMT
     hindi_trigram_lm.json    # Trigram LM for Toy SMT
     dictionary.json          # English-Hindi dictionary

  src/                         # Source code
     __init__.py
     bleu.py                  # BLEU implementation (from scratch)
     toy_smt.py               # Toy SMT system
     moses_interface.py       # Moses decoder wrapper
     word_by_word.py          # Baseline translator
     utils.py                 # Utility functions

  app/                         # Web application
     streamlit_app.py         # Streamlit UI

  tests/                       # Unit tests
     __init__.py
     test_bleu.py             # BLEU tests (pytest)

  scripts/                     # Helper scripts
      hindi_tokenizer.py      # Hindi tokenization utilities
```

Quick Start

For graders without Moses:

```
# 1. Install dependencies
pip install -r requirements.txt
```

```
# 2. Run tests
pytest tests/ -v

# 3. Launch application
streamlit run app/streamlit_app.py

# The app will use Toy SMT and Word-by-Word (Moses optional)
```

For graders with trained Moses model:

```
# 1. Set environment variables
export MOSES_INI_PATH="/path/to/your/moses/model/moses.ini"
export MOSES_BIN_PATH="/path/to/mosesdecoder/bin/moses"

# 2. Install dependencies
pip install -r requirements.txt

# 3. Launch application
streamlit run app/streamlit_app.py

# The app will use all three systems
```

Installation

Prerequisites

- Python 3.10+
- Unix-like environment (Linux/macOS recommended)
- For Moses: perl, g++, make, autotools

Step 1: Python Dependencies

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

Dependencies: - streamlit==1.31.0 - numpy==1.24.3 - pytest==7.4.0 - sacremoses==0.1.1 - pandas==2.2.0 - plotly==5.18.0

Step 2: Moses Installation (Optional but Recommended)

Download Moses

```
# Create Moses directory
mkdir -p ~/moses
cd ~/moses

# Clone Moses repository
git clone https://github.com/moses-smt/mosesdecoder.git
```

```
cd mosesdecoder

# Compile Moses
./bjam -j4
```

Install GIZA++/MGIZA (Word Alignment)

```
cd ~/moses
git clone https://github.com/moses-smt/giza-pp.git
cd giza-pp
make

# Copy binaries
cp GIZA++-v2/GIZA++ GIZA++-v2/snt2cooc.out mkcls-v2/mkcls ~/moses/mosesdecoder/tools/
```

Alternatively, use **fast_align** (faster):

```
cd ~/moses
git clone https://github.com/clab/fast_align.git
cd fast_align
mkdir build && cd build
cmake ..
make
```

Install KenLM (Language Model)

```
cd ~/moses
git clone https://github.com/kpu/kenlm.git
cd kenlm
mkdir -p build
cd build
cmake ..
make -j4

# Add to PATH
export PATH=$PATH:~/moses/kenlm/build/bin
```

Moses Training Pipeline

Overview

Training Moses involves: 1. **Tokenization**: Split text into tokens 2. **Truecasing**: Normalize capitalization 3. **Cleaning**: Remove misaligned/empty sentences 4. **Word Alignment**: GIZA++/MGIZA or fast_align 5. **Language Model Training**: KenLM 6. **Phrase Extraction**: Extract phrase pairs 7.

Model Training: Create moses.ini 8. **Tuning**: MERT/PRO on dev set (optional)

Step-by-Step Commands

1. Prepare Parallel Corpus Create training data files:

```
# Create data directory
mkdir -p ~/moses_training/data

# Example: Create sample corpus
# For real training, use larger corpus (IITB Hindi-English, etc.)
cat > ~/moses_training/data/train.en << EOF
Hello, how are you?
I love programming.
The weather is nice today.
EOF

cat > ~/moses_training/data/train.hi << EOF
  ,      ?


EOF
```

2. Tokenization English:

```
cd ~/moses_training
~/moses/mosesdecoder/scripts/tokenizer/tokenizer.perl -l en \
  < data/train.en > data/train.tok.en
```

Hindi:

```
# Moses tokenizer supports Hindi
~/moses/mosesdecoder/scripts/tokenizer/tokenizer.perl -l hi \
  < data/train.hi > data/train.tok.hi

# OR use Indic NLP Library for better results:
# pip install indic-nlp-library
```

3. Truecasing (Optional for Hindi)

```
# Train truecaser on English
~/moses/mosesdecoder/scripts/recaser/train-truecaser.perl \
  --model truecase-model.en --corpus data/train.tok.en

# Apply truecasing
~/moses/mosesdecoder/scripts/recaser/truecase.perl \
  --model truecase-model.en < data/train.tok.en > data/train.true.en
```

```
# For Hindi, usually skip truecasing or use original
cp data/train.tok.hi data/train.true.hi
```

4. Clean Corpus

```
~/moses/mosesdecoder/scripts/training/clean-corpus-n.perl \
  data/train.true en hi data/train.clean 1 80
```

This creates: - `data/train.clean.en` - `data/train.clean.hi`

5. Train Language Model (KenLM)

```
# Train 3-gram LM on Hindi
~/moses/kenlm/build/bin/lmplz -o 3 \
  < data/train.clean.hi > data/hindi.arpa

# Convert to binary format (faster)
~/moses/kenlm/build/bin/build_binary \
  data/hindi.arpa data/hindi.binary
```

6. Train Translation Model

```
# Using train-model.perl (automates alignment + phrase extraction)
~/moses/mosesdecoder/scripts/training/train-model.perl \
  --root-dir train \
  --corpus data/train.clean \
  --f en --e hi \
  --alignment grow-diag-final-and \
  --reordering msd-bidirectional-fe \
  --lm 0:3:$(pwd)/data/hindi.binary \
  --cores 4 \
  --external-bin-dir ~/moses/mosesdecoder/tools
```

Parameters: - `--root-dir train`: Output directory - `--corpus data/train.clean`: Corpus prefix (without .en/.hi) - `--f en --e hi`: Source and target languages - `--alignment grow-diag-final-and`: Symmetrization heuristic - `--reordering msd-bidirectional-fe`: Reordering model type - `--lm 0:3:path`: Language model (factor:order:path) - `--external-bin-dir`: Path to GIZA++/mkcls/snt2cooc

This creates `train/model/moses.ini` (the trained model).

7. Tuning (MERT - Optional)

```
# Prepare dev set
~/moses/mosesdecoder/scripts/tokenizer/tokenizer.perl -l en \
  < data/dev.en > data/dev.tok.en
```

```
~/moses/mosesdecoder/scripts/tokenizer/tokenizer.perl -l hi \
  < data/dev.hi > data/dev.tok.hi

# Run MERT
~/moses/mosesdecoder/scripts/training/mert-moses.pl \
  data/dev.tok.en data/dev.tok.hi \
  ~/moses/mosesdecoder/bin/moses train/model/moses.ini \
  --mertdir ~/moses/mosesdecoder/bin/ \
  --decoder-flags="-threads 4" \
  --working-dir mert-work
```

Tuned model will be in `mert-work/moses.ini`.

8. Test Decoding

```
# Single sentence
echo "Hello, how are you?" | ~/moses/mosesdecoder/bin/moses -f train/model/moses.ini

# Batch file
~/moses/mosesdecoder/bin/moses -f train/model/moses.ini \
  < data/test.tok.en > data/test.translated.hi
```

Running the Application

Method 1: Using Streamlit Directly

```
streamlit run app/streamlit_app.py
```

The app will open in your browser at `http://localhost:8501`.

Method 2: With Moses Configuration

```
# Set Moses paths
export MOSES_INI_PATH="$HOME/moses_training/train/model/moses.ini"
export MOSES_BIN_PATH="$HOME/moses/mosesdecoder/bin/moses"

# Run app
streamlit run app/streamlit_app.py
```

Using the Interface

1. **Enter Source Text**: Type or paste English sentence
2. **Select Reference**:
   - Choose from 10 built-in examples
   - Upload a .txt file with reference translations
   - Manually enter reference(s)
3. **Translate & Evaluate**: Click button to:

- Generate translations from all available systems
- Compute BLEU scores
- Display detailed n-gram precision tables
- Compare systems side-by-side

4. **Add Custom Translation**: Optionally add your own translation to evaluate

---

Testing

Run Unit Tests

```
# Run all tests
pytest tests/ -v

# Run specific test file
pytest tests/test_bleu.py -v

# Run with coverage
pytest tests/ --cov=src --cov-report=html
```

Expected Test Results

All tests should pass: - Tokenization tests - N-gram extraction tests - Modified precision tests (with clipping) - Brevity penalty tests - Complete BLEU computation tests - Edge case handling

Manual Testing

```
# Test BLEU module
python -c "from src.bleu import compute_bleu; print(compute_bleu('hello world', ['hello wor

# Test Toy SMT
python src/toy_smt.py

# Test Word-by-Word
python src/word_by_word.py

# Test Moses interface
python src/moses_interface.py
```

Moses Validation for Assignment Submission

Use this command before taking final screenshots for the "Moses or equivalent SMT" requirement:

```
python scripts/validate_moses.py --source "Hello, how are you?"
```

Expected: `Validation result: PASS` with non-empty translation output.

BLEU Smoothing (Sentence-Level)

The Streamlit app provides an optional BLEU smoothing toggle: - `Disabled`: strict BLEU (zeros propagate to cumulative BLEU) - `Enabled`: epsilon smoothing for short-sentence comparison

For final reporting, mention which mode was used.

Configuration

Configuring Moses Paths

Option 1: Environment Variables

```
export MOSES_INI_PATH="/path/to/moses.ini"
export MOSES_BIN_PATH="/path/to/moses/bin/moses"
```

Option 2: Edit `src/moses_interface.py`

Modify the `get_default_moses_config()` function:

```
def get_default_moses_config() -> dict:
    return {
        'moses_ini_path': '/YOUR/PATH/TO/moses.ini',
        'moses_bin_path': '/YOUR/PATH/TO/moses'
    }
```

Extending the System

Add More Phrase Pairs:

Edit `data/phrase_table.json`:

```
{
  "new phrase": [
    {"phrase": "      ", "prob": 1.0}
  ]
}
```

Add Training Data:

Edit `data/built_in_corpus.json` to add more parallel sentences.

Retrain Language Model:

Update `data/hindi_trigram_lm.json` with new n-gram counts.

Troubleshooting

Issue: Moses not found

Solution: 1. Install Moses following instructions above 2. Set environment variables 3. Or use Toy SMT fallback (no Moses required)

Issue: Import errors

Solution:

```
pip install -r requirements.txt
export PYTHONPATH="${PYTHONPATH}:$(pwd)"
```

Issue: Unicode errors with Hindi text

Solution: - Ensure files are saved as UTF-8 - Use Python 3 (not Python 2) - Normalize text: `unicodedata.normalize('NFC', text)`

Issue: BLEU score is 0

Possible causes: - No n-gram matches between candidate and reference - Check tokenization (spaces between words) - Verify reference translations are correct - Try lowercasing both candidate and reference

Issue: Streamlit not starting

Solution:

```
# Check if port 8501 is available
lsof -i :8501

# Use different port
streamlit run app/streamlit_app.py --server.port 8502
```

Deliverables Checklist

Part 1 - Task A (8 marks)

- **User Interface (4 marks)**
  - Source text input (textarea)
  - Reference upload/selection (10+ built-in options)
  - Display: SMT output, BLEU scores (1-4 + cumulative), n-gram precision table, BP, lengths
  - Multiple candidate comparison (Moses, Toy SMT, Word-by-Word, Custom)
- **Translation & Evaluation (4 marks)**
  - Moses integration with subprocess interface

  - BLEU from scratch (modified precision, BP, geometric mean)
  - Toy SMT fallback (phrase table + trigram LM + beam search)
  - Multiple reference support

Part 1 - Task B (2 marks)

- **TaskB.md**: BLEU improvement strategies (PDF-ready)

Part 2 - Literature Survey (5 marks)

- **LiteratureReview.md**: Survey on MT evaluation metrics
- **references.bib**: 12+ citations

Code & Documentation

- **src/**: Well-documented Python modules
- **tests/**: Comprehensive unit tests (pytest)
- **README.md**: Complete setup and usage instructions
- **Report.md**: Architecture and design choices
- **SCREENSHOTS.md**: Screenshot checklist (8+)
- **run_checks.sh**: Automated testing script

Moses Training Tips

Using Public Datasets

IITB Hindi-English Parallel Corpus:

```
wget http://www.cfilt.iitb.ac.in/iitb_parallel/iitb_corpus_download/parallel.tgz
tar -xzf parallel.tgz
```

OPUS (Open Parallel Corpus):

```
# Install OPUS tools
pip install opustools

# Download corpus
opus_read -d OpenSubtitles -s en -t hi -w corpus -wm moses
```

Recommended Training Parameters

- **Corpus size**: 50K+ sentence pairs minimum
- **LM order**: 3-gram (balance between quality and speed)
- **Alignment**: `grow-diag-final-and` (best for phrase-based)
- **Reordering**: `msd-bidirectional-fe` (Moses default)

Improving Translation Quality

1. **More data**: Use larger parallel corpus
2. **Better preprocessing**: Proper tokenization for Devanagari script
3. **Domain adaptation**: Use in-domain training data
4. **Tuning**: Run MERT on dev set
5. **Ensemble**: Combine multiple models

Additional Resources

- **Moses Documentation**: http://www.statmt.org/moses/
- **KenLM**: https://kheafield.com/code/kenlm/
- **BLEU Paper**: Papineni et al. (2002) - "BLEU: a Method for Automatic Evaluation of Machine Translation"
- **Hindi NLP**: https://github.com/anoopkunchukuttan/indic_nlp_library