Assignment - Spell Correction for ASR Noun Enhancement

Assignment Report

Name : Aryan Prajapati
Roll no : 2024202009

Contents

Executive Summary

In this assignment, I built a spell correction system to fix errors in medical names that ASR (Automatic Speech Recognition) systems make when transcribing medical conversations. The main focus of this assignment was to correct medical terms, especially drug names, which ASR often misses.

I implemented two different approaches: a baseline model using traditional methods such as Edit distance algorithms and N-gram language models, and an advanced model called T5 which is transformer-based deep learning. Below are my findings:

- The T5 model achieved an accuracy of 82.39% - far better than the baseline's 62.42%
- Word Error Rate dropped from 0.1966 to 0.1654 (lower is better)
- The advanced model was much better at handling complex medical terms that sound different from how they are spelled
- Both models did pretty well at identifying and correcting nouns (around 68-69% recall)
- The improvement shows that the use of context-aware models really helps in medical language correction

1 Introduction

1.1 Problem Statement?

When physicians use speech-to-text systems to dictate their notes, the ASR often mismatches medical terminology. For example, "ibuprofen" might come out as "eye-beu-profen" because the ASR tries to write what it hears phonetically. This is a serious problem in healthcare, where one wrong drug name could be dangerous.

1.2 Solution approach

For this assignment, my approach was:

1. Build NLP based solutions that can automatically correct these ASR mistakes
2. Focus especially on fixing medication names and other medical nouns
3. Try both traditional methods and modern deep learning approaches
4. Compare them properly using different metrics

1.3 About the Dataset

The dataset provided in this assignment has:

- Pairs of sentences - the correct version and the messed up ASR version
- All from medical conversations
- Lots of medication names and medical terms
- Majority of the miss-matches were in the medical terms

2 Understanding the Dataset

2.1 Basic Statistics

Here's what I found when I first looked at the data and ran some error analysis:

Table 1: Dataset Statistics

Statistic	**Value**
Vocabulary Size	9,957 unique words
Average Correct Sentence Length	12.74 words
Average Incorrect Sentence Length	13.03 words

Top 10 Most Common Words:

to, is, for, a, your, medication, and, of, the, used

Most Common Errors Found in Dataset:

- healthcare → health care
- sulphate → sulfate
- vitamin → vitam in
- maleate → malate

Error Category Distribution:

Table 2: Error Types in Dataset

Error Type	Count	Percentage
Word-level/Other	2,637	39.5%
Phonetic/Substitution	2,111	31.6%
Character-level (Likely Typo)	1,858	27.8%
Formatting/Number	103	1.5%
Total Errors	**6,709**	**100%**

2.2 What Kinds of Errors were present?

I found errors of a few types:

2.2.1 Phonetic Errors

These are when the ASR just writes what it hears. Like:

- "ibuprofen" becomes "eye-beu-pro-fen"
- "ZARBERA-N" becomes "Zabara and"

This was actually the most common type of error and also the hardest to fix with basic methods.

2.2.2 Character-Level Mistakes

Sometimes it's just small typos:

- Missing letters
- Extra letters
- Wrong letters

2.2.3 Completely Wrong Words

Sometimes the ASR just picks a totally different word that sounds similar.

2.2.4 Word Splitting Issues

The ASR might break one word into multiple parts or combine words incorrectly.

3 Data Pre-processing

Before building models, I required cleaning up and preparing the dataset.

3.1 Text Cleaning

Cleaning done was:

- Removing special characters from the start and end of strings
- Kept Special characters like /, -, (,) kept them when they were present inside sentence as they can represent important medical abbreviations
- Punctuation - preserved it since it helps with context
- Trim any space at the beginning or end of string
- replace multiple spaces between words with single space
- In standard NLP applications, case are converted to lower case but for this problem I left the case as it is to preserve the medical terms

3.2 Named Entity Recognition (NER)

I used nltk to tokenize and identify Nouns. This helped me to filter out verbs ending with special characters like "cool," as I didn't want to mark then as error since major focus was on medical terms. I also used nltk for Part-of-Speech (POS) Tagging for the same purpose

3.3 Train/Val/Test Split

I split the data as Training, Validation and Test sets. Since there are 10000 rows, this gave:

- 70% for training: 7000 samples
- 15% for validation: 1500 samples
- 15% for testing: 1500 samples

4 Building the NLP Models

4.1 Baseline Model - Traditional Approaches

For the baseline, I combined 2 classic spell correction methods:

4.1.1 Edit Distance (Levenshtein Distance)

This is the classic approach to find candidated for misspelled word - find how many character changes you need to turn one word into another. I:

- Calculated the edit distance between the wrong word and vocabulary to find possible replacement
- Picked at maximum 3 word with the smallest distance
- Used a threshold of 60% which means match must be at least 60%

4.1.2 N-gram Language Model

This approach uses to find probabilities of next word given word or pair or words. This method was used to match the best possible candidates when Levenshtein Distance found many words, all of which were similarly close to miss-matched word. This help us pick the work which from the dataset are most likely to offor after previous word. To keep simple, we use bigram.

4.1.3 (Optional) Dictionary Lookup

From our analysis, we were able to find out incorrect word, either single or in many consecutive words which were miss-spelled and corresponding correct word. This dictionary can be used in many unknown inputs to correct it but in our approach we didn't use it.

Then I used fuzzy matching to find the closest dictionary word to the error.

4.1.4 Baseline Results

Combining all these methods, This is the final result I got in test dataset:

Table 3: Baseline Model Performance

Metric	Score
Word-Level Accuracy	62.42%
Word Error Rate (WER)	0.1966
Character Error Rate (CER)	0.0781
BLEU-1 Score	0.8139
Noun Recall	69.21%

So the baseline was getting about 6 out of 10 words right. Not bad, but definitely room for improvement, especially for those tricky phonetic errors.

4.2 Advanced Model - T5 Transformer Approach

For the advanced model, I used T5-base which is transformer-based architecture. We tried T5-small as well finetuning of which was faster but the final result was slightly worse than T5-base.

4.2.1 Why Transformers?

I chose transformers because:

- They're really good at understanding context
- They can learn complex patterns in how words relate to each other
- They've been shown to work well for text correction tasks
- They can handle long-range dependencies in sentences

4.2.2 Training Setup

These are the hyperparameters I used:

- Learning rate: 2e-5
- Batch size: 32
- Number of epochs: 20
- maximum input length : 128
- maximum target length : 128

Training took 28 mins on Tesla T4 GPU. I monitored the training and validation loss to make sure the model wasn't overfitting or underfitting.

4.2.3 Advanced Model Results

The transformer model did much better:

Table 4: Advanced Model Performance

Metric	**Score**
Word-Level Accuracy	82.39%
Word Error Rate (WER)	0.1654
Character Error Rate (CER)	0.0859
BLEU-1 Score	0.8288
Noun Recall	68.32%

The advanced model got 8 out of 10 words right - a huge improvement! It was especially better at those phonetic errors that the baseline struggled with.

4.3 Comparing Both Models

Table 5: Baseline vs Advanced Model

Metric	Baseline	Advanced	Change	Improvement
Word Accuracy	62.42%	82.39%	+19.97%	+32.0%
WER	0.1966	0.1654	-0.0312	-15.9%
CER	0.0781	0.0859	+0.0078	+10.0%
BLEU-1	0.8139	0.8288	+0.0149	+1.8%
Noun Recall	69.21%	68.32%	-0.89%	-1.3%

What This comparison means:

- **Word Accuracy:** The advanced model improved by almost 20 percentage points! This is huge - it went from fixing 3 out of 5 errors to 4 out of 5.
- **WER:** Lower is better here. The advanced model reduced errors by 15.9%, meaning fewer mistakes in the final output.
- **CER:** This actually got slightly worse (higher). The advanced model sometimes changes characters when fixing bigger word-level errors. I think this trade-off is worth it since word-level accuracy matters more for medical documentation.
- **BLEU Score:** This measures how similar the corrected text is to the correct version. The improvement shows the advanced model produces more natural corrections.
- **Noun Recall:** Both models did about equally well at catching nouns, which was the focus of the assignment. The slight drop in the advanced model might mean it's being more careful about when to make corrections.

5 Detailed Evaluation

5.1 Understanding the Metrics

Let us break down what each metric actually measures:

5.1.1 Word-Level Accuracy

This is straightforward - what percentage of wrong words did the solution fixed correctly?

- Baseline Model: 62.42% (got about 6 out of 10 right)
- Advanced Model: 82.39% (got about 8 out of 10 right)

5.1.2 Word Error Rate (WER)

This counts how many words are still wrong after correction. Lower is better.

- Baseline Model: 19.66% of words still have errors
- Advanced Model: 16.54% of words still have errors

5.1.3 Character Error Rate (CER)

Similar to WER but it looks at individual characters.

- Baseline Model: 7.81% character error rate
- Advanced Model: 8.59% character error rate

The slight increase here isn't great, but it might be because the advanced model sometimes adds or changes characters when it's trying to fix the overall word.

5.1.4 BLEU Score

This compares sequences of words to see how similar they are. It's commonly used in translation tasks.

- Baseline Model: 0.8139
- Advanced Model: 0.8288

Higher is better, and the improvement shows that the advanced model produces corrections that flow more naturally.

5.1.5 Noun Recall

Since the assignment focused on nouns, this measures what percentage of nouns were handled correctly.

- Baseline: 69.21%
- Advanced: 68.32%

Both models did pretty well here. The slight drop in the advanced model might actually be good as it could mean the model is being more selective to avoid incorrect changes.

5.2 Error Analysis - What's Results are still Wrong?

Even with 82% accuracy, the advanced model still makes mistakes. Here's what I found:

5.2.1 Problems the Baseline Had

The baseline model struggled with:

- **Phonetic spellings:** When "ibuprofen" becomes "eye-beu-pro-fen", the edit distance is huge, so the baseline couldn't find the right correction.
- **No context:** The baseline just looks at individual words. So if there are multiple possible corrections, it can't use the surrounding words to figure out which one makes sense.
- **Compound medical terms:** When the ASR splits a medical term into multiple parts, the baseline treats each part separately and can't put them back together correctly.

5.2.2 Problems the Advanced Model Still Has

Even though the advanced model is much better, it still messes up on:

- **Really rare medical terms:** If a medication name wasn't in the training data much (or at all), the model might not know how to fix it.
- **Being too cautious:** Sometimes the model seems to play it safe and doesn't make a correction when it should. This is why noun recall dropped slightly.
- **Character-level issues:** While fixing big word errors, it sometimes introduces small character mistakes.

6 Challenges Faced

6.1 Data Challenges

6.1.1 Challenge 1: Phonetic Errors Are Hard

The biggest challenge was dealing with those phonetic spellings. When "ibuprofen" becomes "eye-beu-pro-fen", it's very different, so much that traditional methods just can't handle it.

How It was dealt with: The transformer model helped a lot here because it learns patterns from the training data. It saw lots of examples of phonetic errors and learned how to map them back to correct medical terms.

Result: The advanced model's 82% accuracy (vs 62% baseline) shows this approach worked well.

6.1.2 Challenge 2: Medical Dictionary Coverage

Medical terminology is vast, and my dictionary couldn't cover everything - especially rare medications and specialized terms.

How It was dealt with: Instead of relying only on dictionary lookup, I used NER to identify medical entities and used the transformer model learn from context.

Result: Both models maintained good noun recall (68-69%), showing this helped.

6.1.3 Challenge 3: Inconsistent Error Types

The errors were all over the place - sometimes just one wrong letter, sometimes completely phonetic spellings.

How I dealt with it: The transformer model was perfect for this because it learned to handle different error types from the training examples.

Result: WER dropped from 0.1966 to 0.1654, showing better handling of diverse errors.

6.2 Technical Challenges

6.2.1 Challenge 1: Preparing Dataset

The data set was medical specific so standard NLP pre-processing cannot be directly applied without inspecting data

- Standard pre-processing like lowering string could not be used as it will not preserve medical term which NLP algorithm needs this. This made me to modify the code to track work when ever their lower form was used in NLP algorithms
- Standard NLP pre-processing of removing special characters cannot be applied in all cases here as they can be part of medical term itself
- There were few miss-spelled word in the non-medical word as well. They were avoided by using POS and NER to identify nouns as most of them were verbs or adjectives.

6.2.2 Challenge 2: Training the Transformer

Training transformer models takes a lot of computational power and time. I had to:

- Use cloud GPU resources (Tesla T4)
- Fine-tune carefully to avoid overfitting and underfitting
- Training taking long time making trying various hyperparameter time and resource consuming

7 Overall Approach

Let me summarize how I tackled this assignment:

7.1 My Methodology

1. **First, I understood the data:**
   - Looked at the sentence pairs
   - Identified error patterns
   - Categorized medical terms
2. **Then, I preprocessed everything:**
   - Cleaned the dataset
   - Applied NER and POS tagging to find nouns
   - Split into train/val/test dataset
3. **Next, I built the NLP models:**
   - Started with baseline (edit distance and n-grams)
   - Moved to advanced (transformer T5 base)
   - Iteratively improved both
4. **Finally, I evaluated:**
   - Used multiple metrics
   - Analyzed errors
   - Compared approaches

7.2 Key Decisions I Made

7.2.1 Decision 1: Focusing on Nouns

I decided to focus specifically on noun correction because:

- Medical terms are mostly nouns
- Medication names are crucial to get right
- The assignment specifically asked for noun focus

This worked well - both models achieved 68-69% noun recall.

7.2.2 Decision 2: Using Transformers

I chose transformers for the advanced model because:

- They're great at understanding context
- They can learn complex patterns

The 20 percentage point improvement validated this choice.

7.2.3 Decision 3: Building a Strong Baseline

I made sure to implement a proper baseline combining multiple traditional methods, not just one simple approach.

8 Summary of Results

8.1 Overall Performance

Here's the complete comparison:

Table 6: Final Performance Comparison

Metric	Baseline	Advanced	Improvement
Word Accuracy	62.42%	82.39%	+32.0%
WER	0.1966	0.1654	-15.9%
CER	0.0781	0.0859	+10.0%
BLEU-1	0.8139	0.8288	+1.8%
Noun Recall	69.21%	68.32%	-1.3%

9 Conclusion

9.1 What I Achieved

In this assignment, I successfully:

1. Built a complete spell correction system for medical ASR errors

2. Implemented both traditional baseline methods and a modern transformer-based approach
3. Achieved 82.39% word-level accuracy with the advanced model - a 32% improvement over baseline
4. Demonstrated that context-aware models work much better for complex medical terminology

9.2 Limitations

There are still some issues:

1. **Not perfect:** 82% is good, but 18% of words are still wrong. For medical use, this needs human oversight.
2. **Character accuracy trade-off:** CER went up slightly. This is worth it for better word accuracy, but it's still a limitation.

9.3 What Could Be Done Next

If I had more time or was continuing this project, here's what I'd try:

1. **Bigger medical dictionary:** Add more medication names from pharmaceutical databases like RxNorm or DrugBank.
2. **More training data:** Especially for rare medical terms and specialized subdomains.
3. **Experiment with larger model:** Since number of model parameters have big affect on result quality, experimenting with larger model could have provided better result but it take more fine tuning time and resource.

9.4 Real-World Applications

This kind of system could be useful for:

- Electronic Health Record (EHR) systems - cleaning up dictated notes
- Helping medical transcriptionists work faster and accurate
- Quality assurance for medical documentation

The 82% accuracy means it could significantly reduce manual work, though you'd still want a human to review the output, especially for critical medical documents.

A Code Structure

Here's how I organized my code: project

- Spell_Correction_for_ASR_Noun_Enhancement_assignment_dataset.xlsx : Provided dataset
- BaseLine Model.ipynb : Training and testing baseline model
- Advanced Model.ipynb : Training and testing Transformer T5 model
- Requirements.txt : Python dependencies