Apertium - Phrase-based MT system

Himanshu Choudhary

Mentors : Rachit Bansal, (supported by Christian Chiarcos)

Google Summer of Code 2022

Contents

1. Profile Information

Name: Himanshu Choudhary

Email: himanshu.dce12@gmail.com

Github: https://github.com/himanshudce

LinkedIn: https://www.linkedin.com/in/himanshudce

Location: Barcelona, Spain

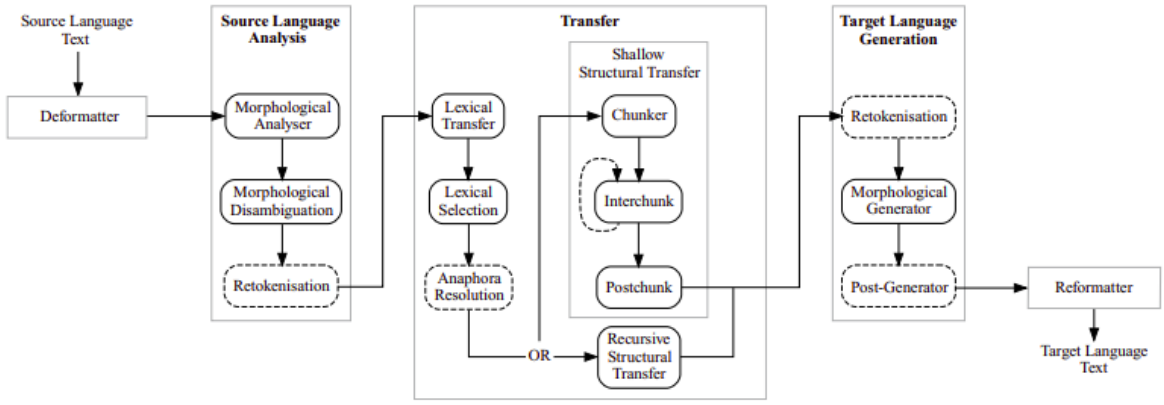
2. Project Overview

The previously developed Machine Translation systems for Sumerian face sparsity issues because of their low-resource. The advanced translation techniques work better with the higher availability of data, whereas for low-resource languages, Rule Based/Symbolic Machine translation is a good alternative which can be used to improvise the accuracy. The objective is to build a sumerian-english machine translation system using one of the widely used machine translation platforms **Apertium**.

3. Implementation Overview

In order to build the Apertium Based Machine translation System We need to build the following pipeline. I built the basic prototype based on the Apertium Based Machine Translation System according to the following pipeline.

Link - https://github.com/himanshudce/Phrase-Based-MT-Sum-Eng



1. Analysis and Pre-Processing

The most time consuming and the most important part, The whole machine translation model is based on the following dictionary and rules preparation -

- Collecting (going through) linguistic resources as mentioned below in each section for the dictionaries of both sumerian and english
- Building morphological dict for both sumerian and english
- systematically comparing the source and the target languages in order to understand what structural changes exist between them.
- Building transfer rules from sumerian to english
- collecting monolingual and bilingual corpora for testing and comparing/analysing rules
- Majorly we need the help of language experts to make correct and precise rules

Three dictionaries to build the system:

A. The morphological dictionary for the language Sumerian:

This contains the rules of how words in the sumerian are inflected (morphology of words and rules). It will be called: **apertium-sum.sum.dix**

Implementation -

a. Define dictionary similar in the similar format of apertium

b. Define alphabet sumerian

A simple vocabulary of character that we can get by iterating over all the Sumerian corpus(UrIII corpus, Administrative text and mtaac_gold_corpus) and saving in set.

1. https://github.com/cdli-gh/mtaac_gold_corpus
2. https://github.com/cdli-gh/Semi-Supervised-NMT-for-Sumerian-English/tree/master/dataset

c. Define symbols/tags (noun, verb, singular, plural, ….. All syntactics)

Define all the possible morphological tags that are possible for a Sumerian Word. Just defining in xml form like below. It will be a simple python script with manual checks place in Sumerian Dict using below sources.

<sdef n="pn"/> is a personal noun.

1. https://github.com/cdli-gh/CDLI-CoNLL-to-CoNLLU-Converter/blob/master/cdliconll2conllu/mapping.json
2. https://cdli-gh.github.io/guides/guide_tagsets.html

d. Add words, lemma, paradigm

For this I will be using mtaac_gold_corpus. So, all the transformation that are present there will be placed in the desired XML format in Sumerian Dict using a python script. The further processing will be done with Sumerian language experts from CDLI.

1. https://github.com/cdli-gh/mtaac_gold_corpus

e. Compile using the Apertium platform as mentioned on the github below
f. Test

Testing by translating multiple sentences from our URIII dataset and checking if we are getting the correct morphology for the whole sentence.

1. https://github.com/cdli-gh/Semi-Supervised-NMT-for-Sumerian-English/tree/master/dataset

The basic Example of dict - https://github.com/himanshudce/Phrase-Based-MT-Sum-Eng/blob/main/apertium-sum.sum.dix

B. The morphological dictionary for the language english:

This contains the rules of how words in English are inflected.

It will be called: **apertium-eng.eng.dix**

Implementation -

a. For English we can use pre-build resources from Apertium itself. We can clone it and use it directly.
    1. https://github.com/apertium/apertium-eng

The eng dict will be modified to contain all the words that we have for the sumerian. To do this I will scan over bilingual dict of sum-eng and will search it in eng dict, words that are not found can be added manually in a dict by ourselves.

The basic Example of dict - https://github.com/himanshudce/Phrase-Based-MT-Sum-Eng/blob/main/apertium-eng.eng.dix

C. Bilingual dictionary:

contains correspondences between the lemma of the words along with their tags(noun, adj,verb,etc) in the two languages.

Like - `<e><p><l>dumu<s n="n"/></l><r>child<s n="n"/></r></p></e>`

`Dummu in English is child and is a noun(n).`

It will be: **apertium-sum-eng.sum-eng.dix**

Implementation

a. Define basic skeleton/syntax as mentioned for other languages like - Noun in one section, Verb in another section, etc.

b. Will use mtaac_gold_corpus to extract the lemma of sumerian word with XPOSTAG and will search its lemma in the above created English dict and put the translation between words simply in one line - sumerian to english

   Like -

   1. https://github.com/cdli-gh/mtaac_gold_corpus

c. If the above dictionary will be unable to capture many words, I will try to implement a Word **alignment model** for Sumerian - English. There are few old prebuilt models (fast_align by IBM link below) which i will apply on our URIII parallel corpus. Another way will be to use/create dictionaries using unsupervised way of machine translation. We can extend the back translation part implemented by rachit in semi supervised translation and get the word alignment dictionaries in OpenNMT (link below). **This will be a bonus task and will be extended after the whole pipeline is ready.**
   1. https://github.com/clab/fast_align
   2. https://opennmt.net/OpenNMT-py/FAQ.html#raw-alignments-from-averaging-transformer-attention-heads
d. Compile in both ways -> left to right, and right to left (sumerian to english and english to sumerian)
e. Test as mentioned below github repo

The basic Example of dict - https://github.com/himanshudce/Phrase-Based-MT-Sum-Eng/blob/main/apertium-sum-eng.sum-eng.dix

Transfer Rules files to build the system:

D. language sumerian to language english transfer rules:

This file contains rules for how the language sumerian will be converted into English language

It will be called: **apertium-sum-eng.sum-eng.t1x**

Implementation -

A. After using sumerian morphological dict to convert each word to its lemma along with morphology we use transfer rule component
B. This is the portion which will take most of the time and I may need dedicated time with someone who is expert in the field to make the precise and correct rules of transformation
C. To build this I will define sections , groups, and attributes which will match the pattern and transform the lemma from sumerian to English. For example - all the Personal nouns will be converted to singular nouns in English with their translation. (check github link for example)
D. I will go through the following resources in order to build the rules, but it may take time in order to understand the language.
    1. https://cdli.ucla.edu/pubs/cdlp/cdlp0002_20160104.pdf
    2. https://www.google.com/url?q=https://kupdf.net/download/hayes-a-manual-of-sumerian-grammar-and-texts_590c1ec5dc0d60651e959e8b_pdf&sa=D&source=docs&ust=1649982282537101&usg=AOvVaw1AdGq14IVxT2NG_LDl_-7d
    3. More resources in future
    4. A Rule, For Example, which I got from christian chiarcos - {d}Nanna[DAT] ... Ur-{d}Nammu[ERG] ... é-a-ni[ABS] mu-na-dù[V]. lit. "For Nanna, Ur-Nammu his.temple built".

       Transfer Rule - [DAT] => "for X", [ERG] => subject, [ABS] => object, [V] => verb. Rules for word order (English): subject verb object for-PP. result: "Ur-Nammu built his temple for Nanna".

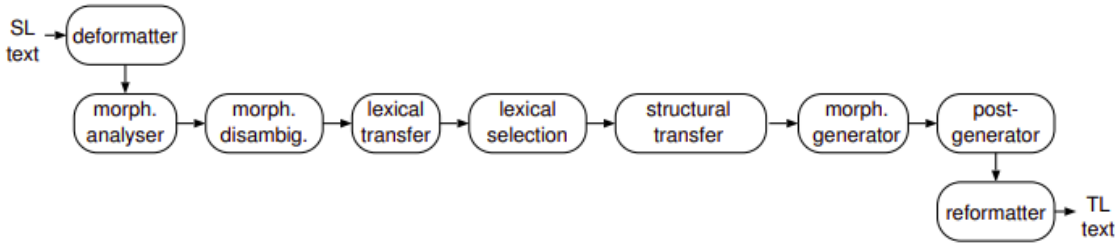
The basic Example of rules - https://github.com/himanshudce/Phrase-Based-MT-Sum-Eng/blob/main/apertium-sum-eng.sum-eng.t1x

E. language English to sumerian language transfer rules (If needed):

This file contains rules for how English will be changed into language sumerian. It is not the part of my current project but

It will be called: **apertium-sum-eng.eng-sum.t1x**

2. Detailed Flow and Integration



Reference: - http://archive.sciendo.com/PRALIN/pralin.2017.108.issue-1/pralin-2017-0022/pralin-2017-0022.pdf

The above diagram represents the detailed flow of the pipeline. To create the dictionaries and pipeline we will follow the below ideas.

A. Deformatter

I will provide the normal text and the Apertium encapsulates any formatting (e.g. HTML or XML tags etc.) information in the input stream. It is basically a tokenization for the sumerian language.

B. Morphological Analyser

For each surface form (lemma) in the stream, returns a sequence of possible analyses. It will be done using the monolingual sumerian dictionary. After passing a deformatter string from the morphological analyser, each word/lemma will be checked in **apertium-sum.sum.dix** and return an analysed string. This we will do simply by passing the text through **apertium-sum.sum.dix (sum-eng.automorf.bin, generated from the sum dict)** which gives all possible output associated with that word.

C. (Morph disambig.) Part-of-speech tagger

Out of all possible analyses for the given word POS will return the most probable analysis. It will be done using our previously built POS tagger system for sumerian language

Sources -

1. https://github.com/cdli-gh/Sumerian-Translation-Pipeline/tree/master/POS_Models. It is based on the HMM (hidden Markov model or Conditional Random field).
2. I will try to integrate the best model to the pipeline if Apertium has functionality to add our own POS tagger. **This will be a bonus task as it may take time to integrate external models with the Apertium base platform.**
3. Otherwise I will use the POS tagger provided by the Apertium itself (Hidden markov model) simply training through the unix interface like given below -
    1. https://wiki.apertium.org/wiki/Tagger_training
    2. And tagged data we already have from our previous project (mtaac scraped) - https://github.com/cdli-gh/Sumerian-Translation-Pipeline/tree/master/Dataset
    3. Another dataset - manual annotations where available, dictionary-based annotations where available, POS model annotations otherwise.

       https://www.google.com/url?q=https://github.com/cdli-gh/mtaac_cdli_ur3_corpus/tree/master/ur3_corpus_data/annotated/comm_conll&sa=D&source=docs&ust=1649982282639080&usg=AOvVaw2LuOGfpeuJjOqJOjNX1hTf

D. Lexical transfer

Here I will use our bilingual dictionary as mentioned above, in that dictionary all words/lemmas are looked up in the bilingual dictionary, and translated. Here I will use (sum-eng.autobil.bin**, generated from apertium-sum-eng.sum-eng.dix)** to lookup and translate from sumerian to english with all possible results.

I. Lexical Selection

To implement this I will use the rules as mentioned above (**apertium-sum-eng.sum-eng.t1x)** to choose the most adequate translation in the English language. We just need to run the file sum-eng.t1x.bin and it will provide us with the best suitable translation. In case of multiple generated sentences Apertium uses maximum-entropy lexical selection models. So apart from normal training we need to generate statistical rules using apertium. We can generate rules either with parallel or monolingual data.To implement this, since we already have data sources I will train it on Apertium predefined model as mentioned here.

1. https://wiki.apertium.org/wiki/Lexical_selection

J. Structural Transfer

Structural transfer also uses the same rule file that we created above. After getting lexical selection, the Apertium system applies common operations including insertion, deletion and substitution of lexical units, for e.g. gender, number and case, etc. Here We need to take care

of word orders, or adding removing verbs within the sentence. For now I am not well aware of how and what structural differences are there in between sumerian and english, during translation what are the things we need to take care of. This will be done with the help of language experts, to write domain specific transfer rules in order to translate the word order/ context form into the correct structure. Here We may need to use recursive transfer to reduce the no of rules/ currently not much aware of how exactly I can use this.

1. https://wiki.apertium.org/wiki/Recursive_transfer

K. Morphological generator

For this i simply have to use our above generated morphological english dict, translating from right to left using `sum-eng.autogen.bin` **(generated from apertium-eng.eng.dix)**

L. Postgenerator

Combining the word, basically orthographic operations. For this I will directly use the prebuilt dictionary for English post processing by Apertium itself. (as mentioned below)

1. https://github.com/apertium/apertium-eng/blob/main/apertium-eng.post-eng.dix

M. Reformatter

This is done by Apertium interface it self, simply converting final output to text string. Just gives the final form.

4. Project Timelines

1. Community Bonding (May 20 – June 12, 2022)

- Getting acquainted with the organisation, interacting with the mentors, and getting to know their desired way of working in the coming weeks.
- Learning more about Sumerian and Apertium.
- Set up Repo and git for the future work

2. Week 1 (June 13 - June 19)

- extract sumerian vocabulary and morphology from mtaac_gold_corpus.
- Extracting tags from above mentioned sources
- discussion with language experts to understand word representation

3. Week 2 (June 20 - June 27)

- writing a code to transfer above generated words and morphology to xml (Apertium format)
- Integrating Sumerian morph dict with Apertium
- testing the pipeline and Modifying rules based on results

4. Week 3 (June 28 - july 3)

- Start working on an English dictionary
- extracting proper Noun data from sumerian pretagged dictionary(which was used before containing proper noun) and the above mentioned POS tagged data and building the english dict
- integrating with Apertium and testing english monolingual dict

5. Week 4 (July 4 – July 10)

- work on building sum-eng dictionary that is required for lexical transfer using mtaac_corpus.

- If we do not get enough translation dict vocab will work on word allignemnts uning URIII source and the models as mentioned above
- Start exploring the rules required for lexical selection from sumerian to english

6. Week 5 (July 11 - July 17)

- continuing gathering and exploring the rules
- building rules with the help of language experts and writing in xml apertium format,
- training the Apertium to auto generate rules using monolingual and bilingual corpora
- Integrating rules with apertium, testing and modifying codes

7. Week 6 (July 18 - July 24)

- Exploring the rules for structural transfer, structural transfer from sumerian to english
- building recursive transfer using the same sources, and again building rules with language experts
- writing rules in apertium html format
- integrating with apertium, testing and modifying codes
- Documenting the work done so far

First Evaluation (July 25 - July 29)

- Basic Pipeline code submission
- Manually **testing** the work that has been done so far.

8. Week 7 (August 1 - August 7)

- building a new / modifying part of speech tagger in the pipeline either using Apertium or using/improvising old POS tagger
- Integrating POS tagger in the Apertium pipeline

9. Week 8 (August 8- August 14)

- Integrating all the components of the whole pipeline along with a post generator on the English language side.
- Testing the integrating pipeline
- Further Modifying the pipeline component based on results like (word allignemnts, recursive transfer model etc.)

10. Week 9 (August 15 – August 21)

- continuing the work from previous week
- Generate possible test matrix for the translation (Word error rate, BLEU, etc)
- We may need to modify and fix components based on the test results

11. Week 10 (August 22 - August 28)

- **Documenting** the work done so far.
- Start working on integrating the pipeline with the framework

12. Week 11 (August 29 - September 3)

- Continuing the integration work,
- Completing the **Documentation** of the work done
- Work on bonus task **if time permits**

Final Evaluation (September 5 - September 12)

- Manually **testing** and finishing details for final submission
- Updating all the work done in the blog.
- Translating whole UrIII corpus

5. Deliverables

Week	Deliverables
Week 1	• Extracted data for sumerian morphological dict and word representation
Week 2	• Integrated sumerina morphological Dictionary With Apertium
Week 3	• Integrated English Morph dict
Week 4	• Integrated English sumerian bilingual dict
Week 5	• Integrated Language transferred rules
Week 6	• Integrated Structured transfer rule
First Evaluation	• Manual test for rules and morphology • Expert Evaluations and Modification
Week 7	• Integrated Part of speech tagger with Apertium pipeline
Week 8	• Integrated Post Generator rules and modified/fixed pipeline
Week 9	• Full integrated pipeline • Evaluation matrix
Week 10	• Result Analysis
Week 11	• Framework integration • Work on bonus task if **time permits**
Final Evaluation	• Testing and Translating whole URIII corpus • Evaluation Matrix • Finalised code and Readme

6. Tech-Stack and Skills

- Bash
- Python
- XML
- Git
- NLP
- Linguistic

7. Why I AM BEST SUITED FOR THE PROJECT

I started my open source contribution journey just 2 years back and contributing to CDLI gave me a great exposure towards working with an open source community, which I am eager to continue further. Along With my previous work experiences, I have a very relatable background to contribute to the project. Since the previous work of sequence tagging will also be used in this project, it makes me more suitable to be part of this project. Also, During the application phase I understood most of the part of the Apertium machine translation platform and am comfortable to apply those techniques to Sumerian language data.

The opportunity to work on this project would be very exciting and would help me to improve my coding abilities and my knowledge in the field of Natural Language Processing.

Contributions

1. **Sumerian Translation Pipeline**

Contributed as a student developer in CDLI, on the GSOC-2020 Project.

https://github.com/cdli-gh/Sumerian-Translation-Pipeline

2. **CDLI-CoNLL-to-CoNLLU-Converter**

Contributed to fix the following issues, where I tried to fix the code break and adding IDs to the head.

https://github.com/cdli-gh/CDLI-CoNLL-to-CoNLLU-Converter/issues?q=is%3Aissue+is%3Aclosed

Experiences

UnitedHealth Group — *Data Scientist*

September 2020-August 2021

Built an information extraction pipeline for legal data extraction and analysis Using Computer Vision, NLP and Graphs for relationship mining

Google Summer of Code (GSOC-2020) — *Intern*

May 2020-September 2020

Worked in the CDLI organisation on building an information extraction pipeline for an ancient language Sumerian (from 3350 BC).

Industrial Technology Research Institute (ITRI), Taiwan — *Intern*

May 2019-August 2019

Worked on an industrial collaborated project regarding Patent knowledge exploration and Concept Discovery by automating the search for a patent's prior art with text similarity and automated Hierarchical Building

Indraprastha Institute of Information Technology(IIIT), Delhi — *Intern*

July 2018-August 2018

Worked as a research Intern in IIIT Delhi and developed a neural machine translation system which performed better than Google translator on Indian language pair (English-Tamil)

Publications

How low is too low ? (ACL_AWS-2020)

Paper on Entity Extraction, language modelling and interpretability analysis for Sumerian language (Proceedings ACL workshop)

Empirical Methods in Natural Language Processing (EMNLP-2018)

Published a paper on Neural Machine Translation for Indian languages as a First Author in **WMT-2018** under **EMNLP** (NMT_2018)

Machine translation for low resourced Indian Languages (LREC-2020)

Published a paper on Machine Translation for low resourced languages in one of the premier NLP conference, LREC (LREC proceedings)

About Me

1. **Education Background**

Erasmus Joint Master Degree Program

ULB Belgium, UPC Spain, TU Eindhoven (TU/e) Masters in Big Data Management and Analytics

Delhi Technological University (Formerly DCE)

Delhi, India

B.Tech - Mathematics and Computing Engineering

2. **Achievements**

1. Awarded Erasmus Mundus Scholarship by European Union

2. Selected within the top 26 research fellows from all over the world in Industrial Technology Research Institute, Taiwan

3. Secured city rank 1 in National Science Olympiad

4. Young Innovation Award at UnitedHealth group

Other commitments During Summer

If time permits I would be more than happy to contribute in cdli-conll/conll-u to cdli-conll converter as well as integration of pipelines to the frameworks. The extended models for pipeline such as POS tag integration and word alignment using ONMT will also be the bonus task and will be completed when the time permits or can be continued after GSOC as well. In the summers I will contribute full time on the project and will devote 30-35 hours per week.