Apertium

Google Summer of Code'20

Proposal

Project: Automatic Post Editing

Contact Information

Name: Saurabh Rai

IRC Nick: srbhr

Location: New Delhi, India

Time Zone: UTC+5:30 (IST)

Email: srbh077@gmail.com

GitHub: https://github.com/srbhr[1]

LinkedIn: https://www.linkedin.com/in/saurabh-rai-9370a0194/ [2]

Who am I? : I'm an Undergraduate Computer Science Student, in 3rd year of college from GGS Indraprastha University, New Delhi. I'm interested in Machine Learning and Natural Language Processing, and always seek to find ways to improve stuff based on them. I love talking about technology, AI, and Cyberpunk 2077.

FOSS Software I have used: My Work always involve FOSS Software and Frameworks, from Python to TensorFlow, from Ubuntu to Arch Linux, I've used many and tried to tweak the software that I use. I have tried to contribute to some of the FOSS Frameworks as well. And I have taken Part in Making some open source projects as well of my own.

Languages I know: Hindi (Native), English (Fluent)

Skills and Knowledge

I'm a Computer Science Student in my 3rd year of College

Languages that I know:

- Python
- C/C++
- Java
- JavaScript

Languages that I'm familiar with:

- Julia
- Bash

Machine Learning, Deep Learning, and Natural Language Processing.

- ML Frameworks that I've worked with: Tensorflow, PyTorch, Spacy, Gensim, nltk, Flair, AllenNlp, openCV etc.
- Data Visualisation and stats libraries: pandas, numpy, seaborn, plotly, matplotlib, scipy etc.
- I've taken Online as well as offline courses to hone up my skills in the field.
- Not only this I've work with projects regaridng the same and I'm currently writing a research paper on Information Retrieval and Extraction
- (Stalled due to sudden college closure coz of CORONA Virus Pandemic).

Courses that I've taken during my college time (Only relevant to project):

- Mathematics (I-IV, it includes statistics, calculus(both), probability and linear algebra)
- Computer Programming
- AI
- Data Structures
- Algorithm Design and Analysis

Why is it that you are interested in Machine Translation?

Machine Translation is one such field that tells us about how tasks like translating languages and serving humanity can be done easily, and not only this, it tells about the scope of how things can be in future, I'm interested in Machine Translation because I'm interested in discovering ways to improve it further and with more features, and the project I'm interested in working with does the same. To improve machine-translation by mining post-edits, and learn based on it.

Why is it that you are interested in Apertium?

Apertium was one of the first Machine Translating Tool invented in the early 2000's in Spain. Till then and since now, it has helped a lot of people translating languages and it's the only Machine Translating Tool/Engine that works offline as well as works with Low Resource languages. Not only this, it's an Open Source and free software for all, and it allows learning students to research and work with them. Working with the Apertium Team(Mentors, and people) is a great opportunity to learn and improve one such great tool to newer accuracy measures and feautures. Not only this, but it also provides resources to learn and research with tools like the Morphological Dictionaries, Transfer Rules, Stream-Parser which can be used to create other tools as well for eg. Automatic Post-Editing tool to create dictionary entries after learning from it. etc.

My Proposal

Which of the published tasks are you interested in? What do you plan to do?

I'm interested in the Project: **Automatic Post-Editing, Mining Post-Editing Dumps (Parallel Corpora) to improve Translation** under the Mentor: Mikel L. Forcada (More Mentors to be added). The Project Aims to find the difference in translation by the Apertium Machine Translation, and the Human Post-Edited and takes appropriate measures and learning algorithms to define the problems/mis-translations for the same and creates the required into information that can be inserted in that Apertium language pair. This information can be either:

- Dictionaries
- Constraint Grammar Rules
- Lexical Selection Rules

Task Description

The main goal of the Project **Automatic Post-Editing, Mining Post-Editing Dumps(Parallel Corpora) to improve Translation** is to create automatically Dictionary Entries (Monodix, Bidix) as automatically and complete by minimizing Post-Editing Dumps(Human Verified Parallel Corpora) to improve translation, and performance of an Apertium Language Pair, a long goal of this will be to automate the process of creating/enriching the Built Language Pairs that are under-incubation.

The Project Consists of two phases :

First Phase

In this phase the Data needs to gathered and converted into specific format, or Post-Editing Operators:

- S : Source Text
- MT(S): Machine Translation of S
- PE(S) or PE(MT(S)): The Post-Edited Sentence (Considered to be accurate)

Now then from this a structured data needs to be created, possibly a Pandas Data Frame (CSV, JSON) for Different Languages.

Second Phase

For this phase the morphological data generated by Apertium's Stream Parser will be used to get the tags/operations of each text. And comparision of the PE and MT of S, will yield some information that can be used to improve the Apertium Language Pair by creating Dictionary, Grammar, Lexical Selection Rules.

Process

The Rationale for this Process is described at [Rationale[3]]

- The First Step is to get the Language Pair data and make it available in the required format (S, MT(S), PE(MT(S))).
- Use the Data with the Appropriate Edit-Distance Algorithms (Explained Later) to find where we need to improve, and get the set of triplets where there is need to improve.
- Then using the streamlined data, our first approach will be to expand the sentences/word's morphological tags using Apertium's Streamparser and convert data into Apertium's Stream format. This will reveal the operators that we are looking for to improve upon.For further easy of use we can store the whole stream or break down the words and store it in the DataFrame.

For Example, Consider the Case for English-Galician Pair:

```
  S: Never engage in action for the sake of reward.
  MT(S): Nunca comprometer en acción para o *sake de recompensa.
  Here sake is not translated.
  We can adapt an approach here to find the missing translation for
*sake from the Parallel Corpus of Post-Edits and try to create an
entry for it
```

Consider another Scenario:

```
  S: Never engage in action for the purpose of reward.
  MT(S): Nunca comprometer en acción para o propósito de recompensa.
  Here for the same sentence we have word "sake"  and it's synonym
"purpose".As "sake" doesn't have a translation pair, whereas
"purpose -> propósito" does.
  We can use this data to create the entry for "sake" to Galician
"ben". Take the example "For the sake of" to "Por ben de." in
English Galician.
```

But this is highly recommended that a Human Post-Edited Parallel Corpus should be available to verify the data.

```
  This example was explained by Mikel:
  S: Los marineros oteaban el horizonte.
  MT(S): The sailors *oteaban the horizon.
  PE(S): The sailors scanned the horizon.
  Here we can do scan.v -> otear.v [or scanned.v -> oteaban.v]
  A morphological guess will make it like Spanish verbs with lemmas
ending in "ar" have "aban" as their imperfect 3rd person plural.
Allowing us to build an entry from here itself.
```

Edit-Distance

In computational linguistics and computer science, edit distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other. Edit distances find applications in natural language processing, where automatic spelling correction can determine candidate corrections for a misspelled word by selecting words from a dictionary that have a low distance to the word in question.

Edit-Distance Algorithms that can be used for this task:

- Levenshtein Distance (The previous work was based on this, hence forth. This will be chosen to work with, below are some other for listing.)
- Damerau-Levenshtein Distance
- Jaro Distance
- Jaro-Winkler Distance
- Match Rating Approach Comparison

For Token Comparision:

- Jaccard Index
- Tversky Index
- Overlap Coefficient
- Cosine Similarity, etc.

The Need for finding the relevant operators over the synonyms, makes us want to compare the tags generated by Apertium.tagger object for the same, and see how many similar elements they have. (While we need to ignore the words that are truly related to the word in general, and take only those elements which are useful from the whole synset.

For example, Take the word sake, it's synset includes:- purpose, determination, aim.
Running morphological analysis on all 4 will yield sake-> noun, but purpose-> noun, verb. aim-> noun, verb. determination -> noun.
for creating a new entry "ben" as a translation of sake in the Galician pair we need to add some data to it, hence by running a Token Comparison, we can extract noun as the common one. (This is just a lay man's example, for the more words, I'm sure we'll get a lot of tags that we need to compare upon.

All these algorithms are available through two Python Libraries (and are optimised).

- Jellyfish[[4]]
- Textdistance[[5]]

The Post-edits and Machine Translations are similar, but only different in sentence length, and some words. The rest 60-70% of the sentence structure remains the same. See the Wikimedia dumps for any language pair on this.

Language Pairs & Data

For testing of the project, I'm working on Available(X) to English or English to Available(X). Our aim is to test this on X to English and English to X as I'm fluent in English and can work with it well. But the longer goal for this is to create this process of discovering pairs Language Free. (The Algorithm for Discovering the Text).

For the Dataset: I'll be using the Available Post-Edits from OPUS Project [6]. (Note: Wikimedia Dumps is a Part of OPUS). (It's FOS Data) Wikimedia dumps are a part of OPUS, and it's available in the provided triplets.)

Work Plan

Community Bonding Period (4th May to 30th May)

Make myself familiar with the Tools, Language Models (Dicts, LUs, Tags etc.), The Howto of Creating Language Pairs, and Anything necessary to the Project.

Create the GitHub Wiki Roadmap and Git Kraken Globoards for the Project and Map its timeline and share with the team.

Talk with the Mentors to get more familiar with them, and their views and improvements about the Project. If the time Allows, start the Project Early.

Week 1-4 (1st June to 29th June)

The Aim for The First Phase

- Get the Data for some Linguistic Pairs. (4-5 Testing Pairs)
- Convert the Data into the required Format and in a Data Frame. (S, MT, PE)
- Use the Edit-Distance Algorithm to find the mis-matches, missing words in Translation.
- Using the Necessary Apertium Libraries, expand the Morphological Dictionary (and Possibly store into its respective dataframe for ease of use later).
- Try to get the synonyms of words from nltk.synset to check if we have something similar available that can be used as well.
- Document the Whole Code.

Deliverable (29th June to 3rd July): A DataFrame Consisting of the Morphological Dictionaries and LUs (Which are later to be used by Algorithm) Note: We're saving time on Edit-Distance Algorithm by using External Libraries.

Week 5-8 (3rd July to 27th July)

The Aim for The Second Phase

- This Phase Focuses Solely on creating the Algorithm that is able to learn post-edits, get the difference between two structure and tags.
- And work upon them to get the data from the Operators and be able to create Dictionary Pairs or Apertium Data and Document the Whole Code.
- This is the most crucial step for the Project; hence a whole work phase of 4 weeks is dedicated to this.
- This whole process is Explained in Task Description (Above [6.2]).

Considering the fact, that a word may or may not have the exact translation in the language, or maybe not in the present form.

```
  S: The twisted cubic is most easily given parametrically as the
image of the map.
```

```
  MT: El va torçar cúbic és més fàcilment donat *parametrically com
la imatge del mapa
  PE: La cúbica torçada es defineix més fàcilment de forma
paramètrica com la imatge de la aplicació.
```

Now if you see, the word parametrically is unknown by MT, hence If we can find words that are beginning with * then we can extract them. (Some work on this is done by Deltamachine in one of her codes, need to extract that function from there. Likely improve that to fit in the new code.) Now, if we align these word triplets. (parametrically, *parametrically, parametrica). We can do a guess by first using the root form of parametric (using external library) and see if that is available in the MT engine. In this case, Parametric is available and can be used to convert, and add into the MT dictionary.

```
  ^parametric/parametric<adj>$^
```

And using this knowledge we can create a starting point for entering parametica into the Catalan pair, as

```
  ^paramètric<adj>$
```

But, as we dive in deep into the problem, we can find such pairs that `^paramètrica/paramètric<adj><f><sg>$` is already available in the MT Engine for Eng-Cat pair, we only need to add some rule about adverbs and their superlative, comparative forms. And given a pair had no translation we could have a starting point where we could have inserted the word. Given another example for this. Take the Galician pair in my proposal page:

```
  S: Never engage in action for the sake of reward.
  MT(S): Nunca comprometer en acción para o *sake de recompense.
  PE(MT(S)): Nunca comprometer en acción para o ben de recompense.
```

Here sake is not translated. But if we try to find the synonym for sake and see if that is available in Galician Pair,

```
  S: Never engage in action for the purpose of reward.
  MT(S): Nunca comprometer en acción para o propósito de recompensa.
```

We get an entry point for ben, from purpose. As purpose is a synonym/similar for sake, and as it's available in both the dictionaries. We can do try a token comparison of sake, purpose, and propósito. And create an entry point for ben in the Dictionary on the basis of matching tokens only. (Not adding anything extra from purpose, only the common tags/matching tags. For that we need to do a token-based comparison, to compare the tags generated by the apertium-tagger.)

The Algorithm is an automation of these tasks itself.

1. 1.To align the text, using L-D and Previous work from Deltamachine as improving upon that.
2. 2.Find out the missing translations, and the mistranslation by comparing from the Extracted Pairs, S, MT, PE.
3. 3.Trying to streamline the above two explained process,
4. 4.Use the code from Deltamachine that she has done for the Bel-Rus pair and try to make it work for more language pairs, Likely it'll be a starting point for working
5. over the algorithm. (Her work is similar to the rationale described by the Mentor Mikel).

Deliverable (27th July to 31st July): Data Generated from the Streamlined Algorithm in the form of monodix, bidix, Grammars, etc.

Week 9-12 (31st July to 24th August)

The Aim for The Third Phase

- Testing of the Data Produced by the Algorithm in Phase Two, and check the accuracy of it.
- Accuracy can tested againg by using Edit-Distance algorithms, as we will have a test set of PE and we'll compare with the improved translation and PE(MT(S)), and try to find our scale at what we have performed.If the MT(S) and PE(MT(S))have less edit rate then, likely it'll be an improvement between the pairs.
- Also, try human verification of the Translation process, by asking someone to volunteer from the Apertium Group to check some language pairs.
- Improve the Algorithm based on this process.
- Documenting the Whole Process.
- Testing the Algorithm onto newer Data and New Language Models
- Trying to shift the Project's Testing of English to other Language Pairs. (This Phase is Experimental)
- If the Time Allows, think of newer ways/ideas to improve/integrate to the Project. (This involves Discussion with the mentors)

Final Submission (24th August to 31st August): The Whole Code along with the Documentation and the Data upon which it was tested and trained.

A Description of Who and How it will Benefit the Society

This Project's sole aim is to Improve the Language Pairs, by creating Data Elements for them, from Parallel Corpus. These Language Pairs are used by the Apertium Developers to create new translational models and my Automatic Post-Editing task will provide them with the resource in for of Dictionary Elements, Grammers, Lexical Rules,which can aid the process of creating/generating Data for an existing Model Pair, henceforth enriching the dictionary and help it achieve state of the art.

For the Apertium User, this will end up as better translation tool. Making the End User more satisfied, and that's a software's sole purpose to help the end-user.

Why Google and Apertium Should Sponsor it?

The Project I'm working on is an Apertium Integration Idea, Automatic Post-Editing, that is with the help of tools, to make the Process of working with Language Pairs easier by Automatically mining dictionary elements, or Apertium Data. This will in a long run will help enrich the existing pair of Language Tools. Apertium's Developers are sure to get the benefit of such autmation task, while working on a language pair given a Parallel Corpus.

By Funding this Project Apertium will get work done on one of its integration idea, and explore newers automation methods, for creating data. For Google to fund this project, it'll help an Open Source Software that is serving a wide community, help Open Source Development for its ideas and promote FOSS Development and Software in general.

Coding Challenge

The Coding Challenge for this task is completed and was discussed with the Mentor Mikel L. Forcada. He considered the challenge a success/pass.

Code is available here Along with the Necessary Details:
https://github.com/srbhr/Test-Edits

Other Plans Besides GSOC

No Other Plans

I have no other Plans besides GSOC, and I can make sure that I will be able to give in full time (40 hrs. a week) for this. However, one situation that can arise from this can be my exams (maybe I don't know yet) being in the first-phase of the Coding Challenge, due to the COVID-19 Pandemic, and lock-downs. This is a situation that can occur but I'm not sure of this. Else, everything goes just as stated in the Work-Plan. And even in exam time I will be able to give in time for coding (20 hrs a week).