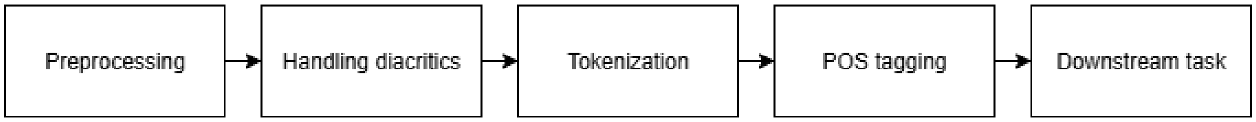
1. Thông tin cá nhân

- Họ tên: Hoàng Thiết Lâm
- Trường học: Trường Đại học Khoa học Tự nhiên, ĐHQGHN
- Email: thietlam04@gmail.com
- Link github bài làm

2. **Xây dựng pipeline xử lý văn bản tiếng Việt bao gồm tokenization, POS tagging, và xử lý dấu thanh điệu. Test với các câu mẫu.**

- To make it suitable for the project of assisting with the disabled, I will analyze 2 to 3 options for each phase of the pipeline, in accordance with the project in mind
- Pipeline:



- Since the questions is just the general "processing" of Vietnamese documents, I will just focus on the preprocessing part first

a. Preprocessing:

- Clean the document in which it comes from many raw sources such as ASR, OCR or speech to text
- Handle specific Vietnam language's issues
- Ensure comprehensibility for later phases of the pipeline
- 1st step: Normalize Unicode and encoding
  + Normalize because Vietnamese has different ways to represent diacritics.
  + Example: à can be a single unicode character or combination "a+`".
  + Tool: ftfy library
  + Required for all Vietnamese NLP tasks so that later phases such as tokenization or POS tagging can work. For example, a tokenizer may not recognize the word "hòa" if it's encoded improperly.
- 2nd step: Lowercasing
  + This can be optional, since Vietnamese words are case-insensitive.

    + Tool: .lower() in Python.
    + Advantage: reduce vocabulary size, especially for larger tasks.
    + Disadvantage: Lose information about named entities. For example: “Thảo Nguyên” and “thảo nguyên”.
    + For a project assisting disabled persons, lowercasing may simplify the output.
- 3rd step: Remove non-word characters
    + Clean up text inputs from special characters(from keyboard input or bugs in the text receiver)
    + Tool: Regex
    + Keep punctuation if sentence segmentation is necessary depending on the tasks. If there is a POS tagging phase later then keep the punctuation in order for the tagger to recognize the role of the words in a sentence(obviously).
    + Remove numbers if it is not semantically important
- 4th step: Sentence segmentation
    + Vietnamese has ambiguous sentence boundaries, especially if the text came from spoken or OCR text
    + Tool: underthesea. Cannot use regex since there can be words with dot notation like “TP.HCM”
    + Useful for downstream tasks such as POS tagging
- 5th step: Spell correction
    + Optional
    + Useful for noisy text with inputs from OCR or speech
    + Tools: VSpell, VnSpell,...
- 6th step: Normalize numbers to words(optional)
- Depending on the task that we want, we can choose all or a subset of a preprocessing techniques I have mentioned

b. Handling diacritics

- Restore correct Vietnamese tone marks in text that lacks diacritics(“Toi muon an com” to “ Tôi muốn ăn cơm”) / incorrect diacritic placement(“hoà” to “hòa”).
- Text can be different based on the accents and tone marks of the diacritics. For example: “ma kia” can be “má kìa” or “ma kìa” or “mã kìa”

- Challenges of this phase
    + Ambiguity. This can be addressed through considering surrounding words
    + Noise from ASR / OCR inputs
    + Input speed / constraints according to the disabled needs
- Rule based techniques: Insufficient for complex or unknown words
- Deep-learning
    + Phrase-based or Seq-to-seq
    + BiLSTM + BiGRU
    + Pointwise character-based (LinearSVM)(Machine Learning
    + Transformer
- However, to take into account the implementation requirement for the disabled which needs fast real-time correction, fast inference time models should be used.

c. Tokenization

- Identify meaningful word boundaries in text, especially for Vietnam language where each word may consist of multiple subwords separated by spaces
- To integrate with downstream tasks like POS tagging.
- Handle noisy input texts
- Challenges:
    + Ambiguity: “sinh viên đại học” are 4 separated words or 2 compound nouns.
    + ASR input: No punctuation, no diacritics: “toi muon an com” is a hard example to tokenize
    + OCR input: Split or joined syllables, special characters
- Tools:
    + Statistical or rule-based like VnCoreNLP, pyvi, underthesea. These are lightweights, no training needed, suitable for fast inference implementation but does not adapt well for unknown words, context ambiguity, compound words
    + BPE tokenization. Commonly used for Transformer-based models. Robust for unknown words, compatible with DL models. However, output is logits, not human-readable so need extra steps

        + ViSoBert for social media text

d. POS tagging

- Match each token with a label representing its part of speech(noun, verb, adjective)
- Challenges:
    + Multi-syllable words
    + Ambiguity: The word “cháy” can be interpreted as adjective or noun depending on the context
    + Limited annotated data
    + Domain and dialect differences: Some words may be different depending on specific dialects(South / North) or domains(medical / social media)
- Tools:
    + Underthesea: Fast and High accuracy, easy to use
    + VnCoreNLP: Also fast and high accuracy, but need java
    + RoBERTa using GPU so slower, helpful for transformer based models
    + CRF models also requires GPU so slower, but very high accuracy
    + PhoBert with very high accuracy but medium inference speed due to GPU
- For our project, we should consider non-gpu tools for POS tagging like underthesea or VnCoreNLP.

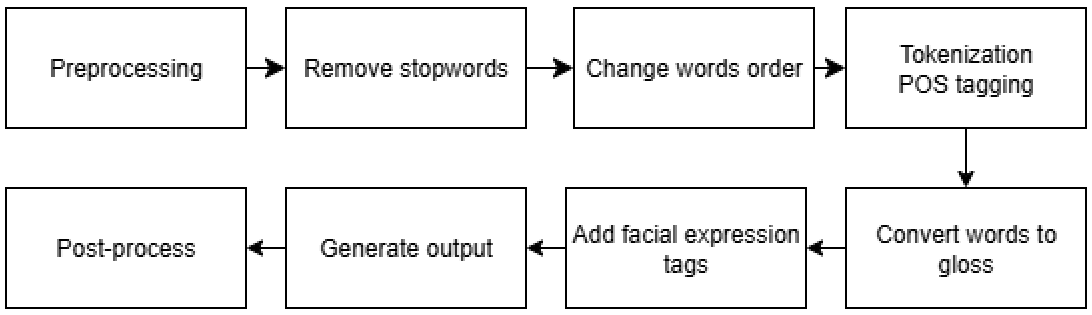
3. Phân tích sự khác biệt về cấu trúc ngữ pháp giữa tiếng Việt và ngôn ngữ ký hiệu. Xây dựng hệ thống chuyển đổi câu tiếng Việt sang cấu trúc ngôn ngữ ký hiệu.

a. Difference of syntactic structure between Vietnamese and Sign language

- Vietnamese syntactic structure
    + Is a language with a word order SVO(Subject - Verb - Object). For example “Tôi ăn cơm”(S: tôi, V: ăn, O: cơm)
    + Contains functional words: đã, đang, sẽ, là, của….
    + Contain punctuations, tenses to represent syntax meaning such as level, question, negativity…
- Sign language syntactic structure:

    + Does not follow general SVO order: Tôi ăn cơm -> Tôi cơm ăn
    + Remove mostly all functional words: Tôi đã ăn cơm rồi -> Tôi cơm ăn
    + Depends on context information such as emotions, structure, space
    + Does not use tense, negativity, question, but use body language

b. Pipeline for conversion from Vietnamese to Sign language



- Goal: Convert a Vietnamese text into simplified sign language text, can be used for animation or…
- 1st step: Preprocessing
    + Objective: Normalize the input to remove noise, segment meaningful units, simplify complex grammar structure
    + Text cleaning and normalization through removing extra whitespaces, special characters… but no lowercasing
    + Sentence segmentation to avoid long nested Vietnamese text
- 2nd step: Remove stopwords
    + Since sign language focus on content-heavy words, removing stopwords improves clarity by focusing on semantically important words and simplifies the sentence structure into “topic-comment” or SOV
    + Define a comprehensive Vietnamese stopwords list(using pyvi or underthesea or VnCoreNLP)
    + Tokenize sentence to recognize stopwords accurately(in terms of compound words like “của mình”)
    + Optional: Track emotionally expressive words like “nhé”, “đấy” to track facial expressions
- 3rd step: Reordering word order

    + Sign language uses topic-comment or OSV or SOV like “tôi ăn cơm” -> “tôi cơm ăn” to enhances visual communication clarity
    + Use underthesea/VnCoreNLP to extract dependency parsing tree
    + Create rule-based ordering to reorder tokens
    + Detect time adverbs and move to the front
- 4th step: Tokenization, POS tagging
    + Objective: Break Vietnamese text into correct tokens that captures enough information for context. Assign POS tags to identify main content words.
    + Tokenization: Use tools like VnCoreNLP/Underthesea for fast inference time. These tools take into consideration syllable-based tokenization, not like English tokenizers.
    + POS tagging: Use tools like VnCoreNLP/Underthesea/pyvi/Stanza with some consideration about accuracy - inference speed. Filter un-needed words of later phases, identifies verbs, nouns, adjectives….
- 5th step: Convert words to sign language glosses
    + A gloss in sign language is a written word that represents a sign in sign language. It’s a label for the sign, typically in uppercase
    + Lemmatization: convert derived word forms such as“đã học” to “học”, “đã ăn” to “ăn”.
    + Build or use Vietnamese gloss dictionary of publicly existing resources or manually build
    + Handle multi-word tokens like “làm ơn”, “đi vệ sinh”(from the previous tokenization process)
- 6th step:
    + Facial expressions are important for both grammatical and emotional functions. This step add sign sequences with facial expression markers that are semantically and syntactically suitable
    + Example: Eyebrow raise/furrow, head tilt, mouth movement, eye squint…
    + Based on POS tagging from previous phases, define rules such as: Negation -> Head_Shake, Topic marking -> Head_Tilt

    + Use sentiment/emotion cues with words such as “vui”, “buồn”, “tức giận”, “lo lắng”....
    + Or define Machine Learning/Deep Learning models to label gloss with expressions.
- Last steps: Generating outputs
    + Goal: Convert the processed gloss sequence into formats which are friendly to the disabled: Animation/Avatar…
    + Output format: Video animation / structured gloss format
    + Textual output is for evaluation - sign language learners
    + Sign avatar animation using graphical tools like Blender/Sign3D/MediaPipe…
    + Deployment on app has to take into consideration speed - accuracy - visualization ability