Documentation

1. Problem overview
   1. Description: Building a knowledge base (KB) for music domain using information from some Vietnamese websites (including <http://nhaccuatui.com> & <https://mp3.zing.vn> & <http://chieu-cao.net>).
   2. Requirements (predicates to be extracted):

* Predicate 1: birthName (Person, Text)
* Predicate 2: birthPlace (Person, Location) with url to the corresponding vi.wikipedia page
* Predicate 3: birthDate (Person, Date/Time)
* Predicate 4: Height (Person, Number) in cm
* Predicate 5: composedBy (Song, Person/Organization)
* Predicate 6: performedBy (Song, Person/Organization)
* Predicate 7: lyric (Song, Text)
* Predicate 8: instrument (Person, Text)
* Predicate 9: profession (Person, Text)
* Predicate 10: memberOf(Person, Organization) (+ temporal info)
* Predicate 11: image (Person, url)
* Predicate : linkToWiki (Person, url) (vi.wikipedia.org)
  1. Expected output: SPARQL endpoint to explore the KB

1. Solution proposal
   1. Pipeline

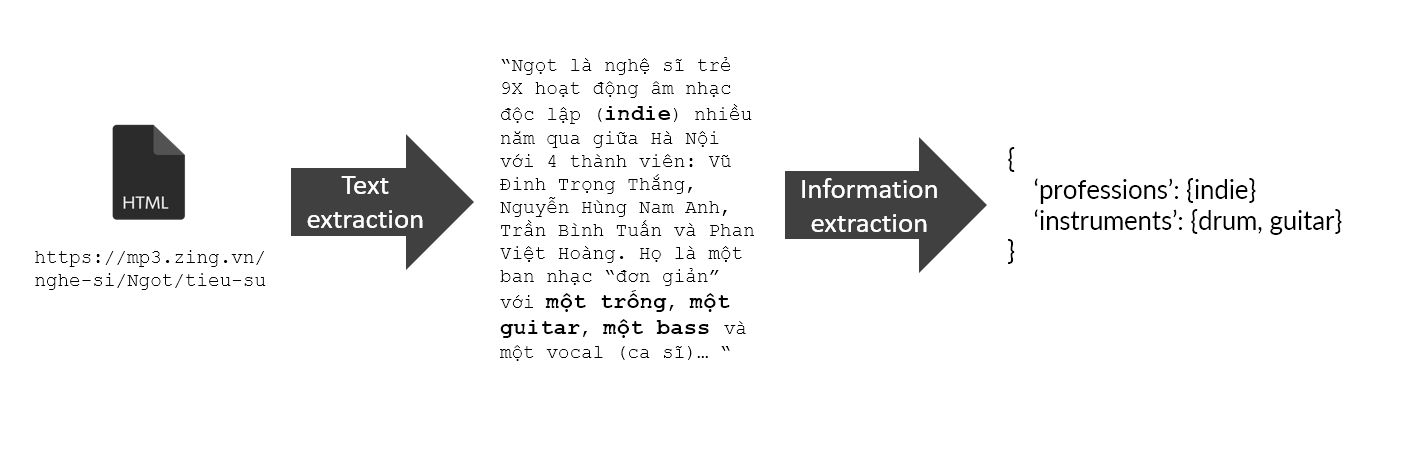
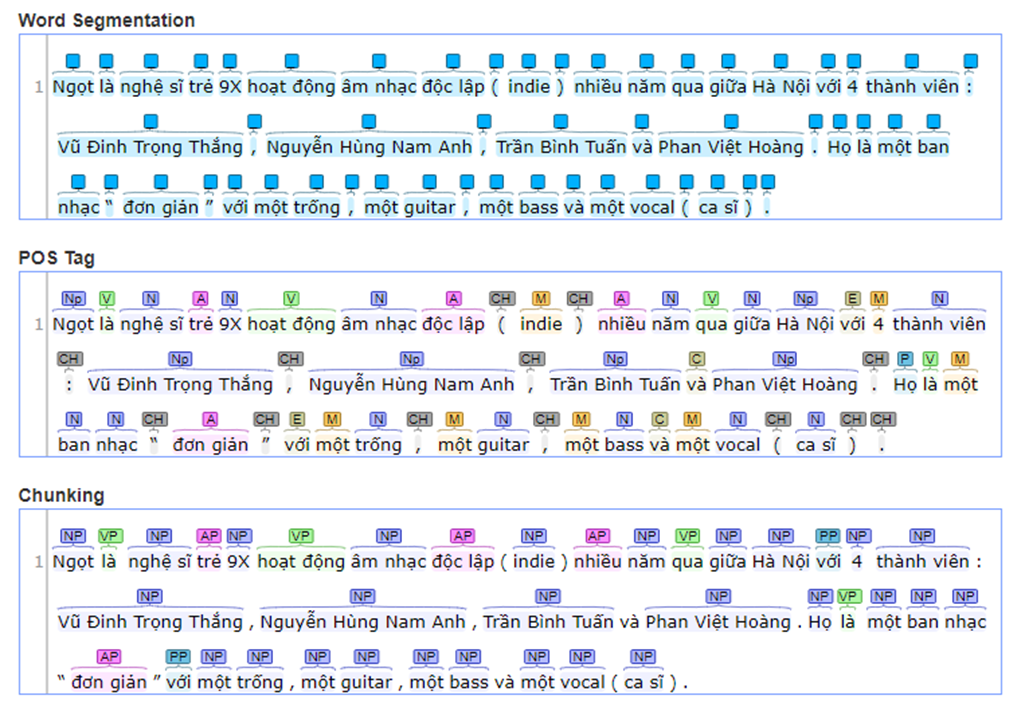
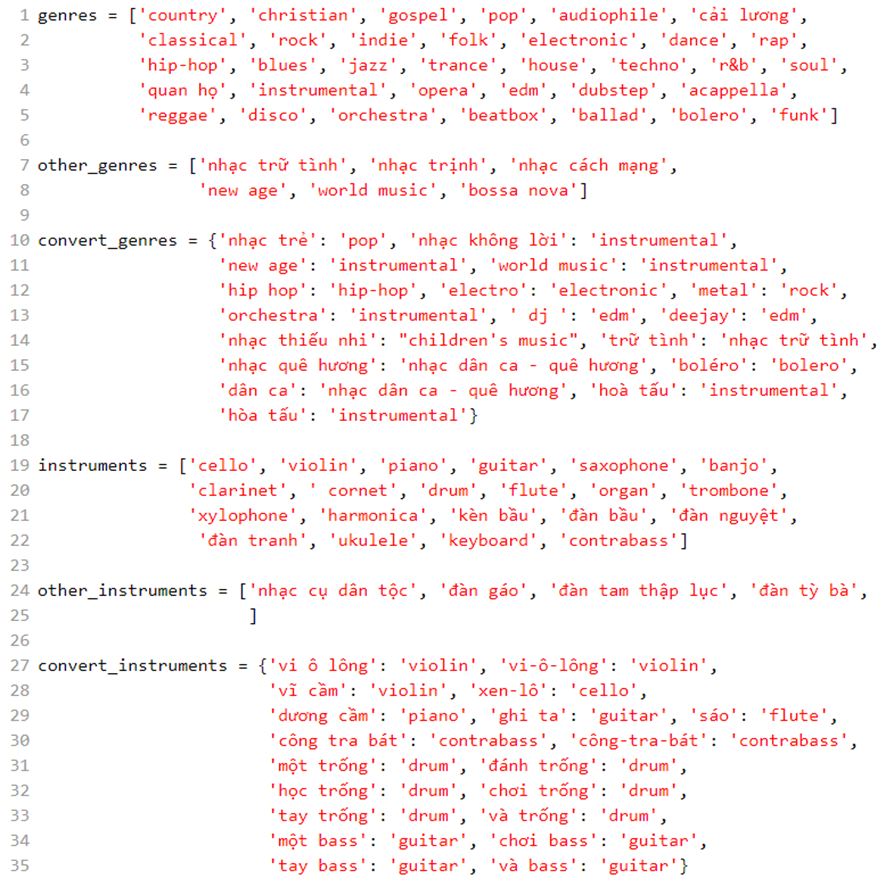
* Crawl necessary pages and store them for later processing
* Extract information from each site
* Combine all the extracted information (information alignment)
* Import to a SPARQL endpoint
  1. Tools
* Programming language: python
* Web scraping: scrapy and some python built-in modules
* Html parsing: BeautifulSoup4
* NLP: underthesea
* Database: Virtuaso

1. Solution details:
   1. Web crawling:
      1. mp3.zing.vn

* Follow all links starting from <https://mp3.zing.vn/the-loai-nghe-si>, only crawl those that ends with ‘/tieu-su’
* Crawl all the song list pages in every biography page (e.g. <https://mp3.zing.vn/nghe-si/JustaTee/bai-hat>, <https://mp3.zing.vn/nghe-si/JustaTee/bai-hat?&page=4>)
* Crawl all the song pages listed in every song list page (e.g. https://mp3.zing.vn/bai-hat/Trong-Rong-JustaTee/ZWZBZBEC.html)
* Might contain some 404 pages
  + 1. nhaccutui.com
* Follow and crawl all artist biography pages starting from <https://www.nhaccuatui.com/nghe-si.html>
* Crawl all the song list pages in every biography page (e.g. <https://www.nhaccuatui.com/nghe-si-Rhymastic.bai-hat.html>, <https://www.nhaccuatui.com/nghe-si-Rhymastic.bai-hat.17.html>)
* Crawl all “Official” song pages listed in every song list page (e.g. <https://www.nhaccuatui.com/bai-hat/treasure-rhymastic.6oKIssEnzPa5.html>)
  + 1. chieu-cao.net
* Follow and crawl all pages starting from <https://chieu-cao.net/category/ca-sy>
* Result are store in a folder structure with the relative path of a .html file is the route of the corresponding URL (e.g. the page <https://chieu-cao.net/thong-tin-tieu-su-tien-cookie.html> can be found at …/chieu-cao.net/thong-tin-tieu-su-tien-cookie.html
  + 1. Storing format (for mp3.zing.vn and nhaccuatui.com)
* Store list of URL in .csv files
* Store html text in to .txt files, at most 10000 pages per file.
* Before every html content, there is a line contains the url of the content and a line contains the delimiter string (“##### NEW FILE #####”)
* Html pages are stored in the same order as the order of their corresponding url in .csv files
* Artist biography pages are stored in files begin with “tieu\_su\_”, song list pages are stored in files begin with “song\_list\_”, song pages are stored in files begin with “songs\_”
* Mp3.zing.vn has 588861 pages which are stored in 3 “tieu\_su\_” files, 4 “song\_list\_” files and 52 “songs\_” files with the total size of 58.5 GB approximately
* Mp3.zing.vn has 150176 pages which are stored in 1 “tieu\_su\_” files, 3 “song\_list\_” files and 15 “songs\_” files with the total size of 26.1 GB approximately
  + 1. Note
* All data can be found at <https://drive.google.com/drive/u/1/folders/1SxL5OSiffIjonv_uUGWZYThHRjSzlJgo?fbclid=IwAR27v7FJaSiwHrIYimzASt9SlEJiY7ohOB4tBaid2TdEt1Q4k3CuygrqTiI>
* Please read the README file (Vietnamese) for more information
  1. Information extraction
     1. Storing format:

* + - 1. Artist data
* Artist data is store in 2 python dictionary objects, 1 for mp3.zing.vn and 1 for nhacuatui.com, let’s call them kb objects.
* The kb’s keys are the URLs of every artist biography collected after the process described in section 3.1
* The kb’s values are dictionary objects containing:
  + ‘name’: string – the name of the artist
  + ‘city’: string – URL of the vi.wikipedia page about the city where the artist was born
  + ‘country’: string – URL of the vi.wikipedia page about the country where the artist comes from
  + ‘birth\_name’: string – the real name of the artist
  + ‘dob’: string of date in dd/mm/yyyy format – the date of birth of the artist
  + ‘height’: int – the height of the artist in cm
  + ‘image’: string – URL to a image of the artist
  + ‘instruments’: set of strings – instruments that the artist can play
  + ‘professions’: set of strings – the artist’s genres (will be renamed to ‘genres’ later)
  + ‘member\_of’: set of strings – name of bands that the artist was and/or is a member of (could include temporal info later)
  + ‘wiki’: string – URL of the vi.wikikipedia page about the artist
    - 1. Song data
* Stores in 2 kb objects like artist data
* The kb’s keys are the URLs of every song collected after the process described in section 3.1
* The kb’s values are dictionary objects containing:
  + ‘name’: string – the name of the song
  + ‘composed\_by’: set of strings - URLs of the biography pages of all the artists that compose the song
  + ‘performed\_by’: set of strings – URLs of the biography pages of all the artists that perform the song
  + ‘lyric’: string – the lyric of the song
  1. Information alignment: TODO
  2. Import to a KB: TODO

1. Information extraction
   1. Predicate 8 (instruments) and 9 (professions)
      1. Input

* Html of artist biography pages from mp3.zing.vn and nhaccuatui.com
* Example:
  + <https://www.nhaccuatui.com/nghe-si-den-vau.html>
  + <https://mp3.zing.vn/nghe-si/Ngot/tieu-su>
* Note: Biography pages from mp3.zing.vn have urls ending with “/tieu-su”, not the overview page (<https://mp3.zing.vn/nghe-si/Ngot>)
  + 1. Pipeline:
    2. Text extractions: using css selectors to extract biography contents
* nhaccuatui.com: ‘#divDescription’
* mp3.zing.vn: requires more than 1 selector due to inconsistent html layout
  + 'body > div.wrapper-page > div.wrap-body.group.page-artist-all.page-artist.container > div.wrap-2-col > div.wrap-content > div > div.row > div > div' (majority)
  + body > div.wrapper-page > div.full-banner > div.wrap-body.group.page-artist-all.page-artist.container > div.wrap-2-col > div.wrap-content > div > div.row > div > div'
  + 'body > div.wrapper-page > div > div.container > div.wrap-body.group.page-artist-all.page-artist.container > div.wrap-2-col > div.wrap-content'
    1. Information extraction
* First attempt: Apply pos tagging and dependency parsing
  + Word segmentation alone is computationally expensive
  + Impractical, cannot scale (having 20000+ samples)
* Second attempt: Build a lookup dictionary manually
  + Time-consuming
  + Cannot cover all cases
  + Need to standardize different terms that have the same meaning
  + Could include false cases
  + Safe choice, a good starting point
    1. Handling duplicates
* There are many artists with the same name from mp3.zing.vn
* Need to disambiguate them so the data can later be merge across sites
* The process cannot be done systematically
* Methodology:
  + Remove artists that have no song
  + If 2 records refer to the same artists, remove the one that has less information (or less song)
  + If 2 artists have same name but different real names, pick the less popular one (or have less song) and modify their record as follow: add “ == [real\_name]” to the field ‘name’ where [real\_name] is that artist real name.
  + If 2 artists have same name, same real name, but come from different countries, pick the less popular one (or have less song) and modify their record as follow: add “ >> [country]” to the field ‘name’ where [country] is the country where that artist comes from.
  + If 2 artists have same name, same real name, come from the same country, but have different major genres pick the less popular one (or have less song) and modify their record as follow: add “ => [genre]” to the field ‘name’ where [genre] is the major genre of that artist.
* During this process, many other information is found (height, dob). Those information has been added manually into the kb object and should not be overwrote.