NLP Project - Ontology of the Three Kingdoms

Roberto Salvaneschi

ro.salvaneschi@gmail.com

Introduction

The aim of the project is model an ontology for the famous Chinese novel *Romance of the Three Kingdoms*, using natural language processing techniques to retrieve information out of the novel itself. The final ontology doesn't have many individuals, but almost all data have been gathered through natural language processing techniques. This relation will explain how to use the code, but most important it will explain how has been developed. Since this relation may be long, if you just want to try the codes look for the "*Play with the Codes!*" section, after making sure to check the "*What you actually need*" part.

Tools Used

Ontology has been created using the last version of *Protégé*[1], which is the 5.5.0, and the final work has been exported as an .owl file.
The NLP-codes, saved as *romance-nlp.ipynb* and *romanceFree-nlp.ipynb,* has been produced with *Python 3* with the help of few libraries: most of them are standard for *Python*, such as *string* and *numpy*, but some like *nltk* may required a separate download. The file *romance-onto.ipynb* was an attempt to interact with *Protégé* via *Python* and to make this possible other non-standard libraries have been imported; still the download of those modules are not mandatory since the experiment in that direction didn't lead to any interesting results.

What you actually need

To appreciate the ontology:

- *Protégé* or some other program that can open .owl file.

To appreciate the codes:

- *Python*, not sure about differences between version 2 and version 3;
- The *nltk* library for *romance-nlp.ipynb* and *romanceFree-nlp.ipynb*;
- The *owlready2* library for *romance-onto.ipynb*;
- Dataset, provided in the folder and explained below;
- Files with list of words, provided in the folder and explained below.

Note that in the folder there are many lists, most of them can be easily recreated with the provided code. Those lists have been created to speed up the work and, in a possible future, been able to deal with greater amount of data which otherwise could be impossible for *Python* to load together. Some lists are not reproducible with the code, those files have been created by hand (sort of) and cleaned with simple operations that have been removed from the final version of the code since they were not related to the project.

Files Explanation

Dataset – Romance of the Three Kingdoms

As already mentioned, the dataset is about the historical Chinese novel and, like for any important books, there are many different translations. At first I didn't care about this, but the Adelaide University eBook Library[2], the site from which I took the first chapter, closed so I had to search for another source. Finding a good alternative required some time due to text-format and file-extension problems. This kind of issues are usually easy to solve, but the size of the file made the solution a little bit complicated; still the greater obstacle was the translation. After a text analysis performed by hand, I decided to use the Brewitt-Taylor version because, among the most recent translations, it is the one where there are less usage of pronouns for addressing characters.
Fun fact: the Brewitt-Taylor version was the one provided by the Adelaide University.

Dataset – What actually is in the folder?
The Dataset folder contains three files:

- *RotTK_FULL(Brewitt-Taylor).txt* → the full novel translated by Brewitt-Taylor;
- *chap001-004.txt* → chapters from one to four, taken from the full novel;
- *chap005-012.txt* → chapters from five to twelve, taken from the full novel.

Loading the entire novel was not possible with *Python* due to file-size so the old-good copy&paste to a new file was the solution. Analysing a small set of chapters at time is good because all data you can retrieve will be somehow limited to a certain set of years and events, making the inferring of certain details and connections a lot easier.

Lists – Set of Words Clustered Together for a Reason

While looking for the proper version of the book, one of those I analysed had two interesting sections: in the first one each main character gets a little description of what he/she had done while the second one gave a little timeline with some described events. The descriptions are way too simple so they can't be used for the ontology, but having all the important names stored somewhere can be useful, so I cleaned and filtered the list to obtain names. I didn't clean the file about events because I decided to focus on characters, but I've upload it anyway for completeness and potential future usage.

Lists – What is in the folder?
The Lists folder contains eight files:

- *Main Characters.txt* → The original characters list with a simple description;
- *AllNames.txt* → List with all names from the file above, without description;
- *menList.txt* → List with male characters.
- *LadyList.txt* → List with important female characters. Basically all of them are wifves or relatives of important male characters. A bit sexist but it was the 160~280 AD so forgive them.
- *AliasesList.txt* → Some characters changed name during the course of the events due to a weird Chinese custom*. Also two characters became Emperor and they get addressed accordingly.
- *WarWords.txt* → List of words related to war. This list has been created with a copy&paste of terms from the internet[3] so many of them are out of context.
- *PoliWords.txt* → List of words related to politic. This list has been created with a copy&paste of terms from the internet[4] so many of them are out of context.
- *Main Events.txt* → Year and name of the event with a little description.

Note that those lists are just starting points, they should be updated whenever possible with, for example, newly discover characters or most-used words related to warriors or politicians. I did that a little with *warWords.txt* and *poliWords.txt* in order to improve metric's precision. Adding other coherent words could improve metric's reliability, specially for the politic's metric which has less words to check, while removing out of context terms will improve efficiency. Last but not least, *menList.txt* may have some female character inside it, further cleaning would be good, but even now errors should be minimal due to the small number of female characters. Thank you sexist old-author!

*Chinese people have three names, two are like ours while the third is called style name. The style name is used instead of the first name when someone reach adulthood. In the novel most characters are already adults and we don't know their 'childhood' name, but some of them are introduced as children so, in order to track all their life, the *aliasesList.txt* stores full-adult name before and childhood ones after.

Codes – Extract informations out of the novel to create an ontology

Romance-Ontology – Building ontology via Python...or tried, at least
Even if this wasn't in the project proposal, with the *romance-onto.ipynb* file I wanted to create owl-individuals via *Python* code in order to generate tons of them quickly, using data gathered with NLP techniques to fill owl-data and owl-object properties. After that, I would have just updated some details by hand through *Protégé* when needed. Unfortunately this experiment failed due to complication in linking ontology with *Python*. Since this was an extra and finding the dataset also took more time than expected, I decided to use the remaining days to produce a better presentation rather than create more individuals by hand...still, it would have been more interesting if performed automatically with *Python* rather then by hand via *Protégé* so I'm sad for not being able to do it.

Romance-NLP – Guided overview of the functionality
The *romance-nlp.ipynb* file has been reordered in such a way that it can be used as a sort of tutorial. Not only there are many comments to make operations easily understandable, but the execution flow indirectly suggests how to analyse the novel, aside from the fact that the user has or not some knowledge about the domain.
The idea is to follow the research about a chosen character, answering suggested questions with already setted functions and then use those informations to enrich the ontology. All deductions are explained in the code in order to give a sort of empiric proof of correctness of results and reliability of the tools. All discovered informations should already have been inserted in the ontology.

RomanceFree-NLP – Retrieve information from text
If the previous file was a tutorial, this *romanceFree-nlp.ipynb* is a demo.
The code has been arranged in such a way that let the user play with the dataset. Comments between cells are suggestions or explanations of what the user should do, helping him/her in exploring the dataset freely while searching and retrieving informations to create individuals for the ontology.

Ontology – Giving Shape to Knowledge

The ontology is the only .owl file in the folder, created with *Protégé* and populated by hand with informations retrieved from the *Python* files above: say hi to the one and only *ontology-of-the-3-kingdoms.owl*! Jokes aside, while searching ontologies online I failed in finding other ones related to this domain so my ontology could actually be the one and only. At least in English language. Further analyses on this can be founded below.

Play With the Codes!

As already mentioned, codes are arranged in such a way to be as clear as possible. First you should run the *romance-nlp.ipynb*, reading comments to understand how NLP has been used for building the ontology. Once you've finished with that you can play with *romanceFree-nlp.ipynb* to have a taste of what it means searching data to fill the ontology. This second file has comments that explain and suggest how to find informations you need.
About *romance-onto.ipynb*, I've left it because it was an interesting idea, but it didn't work so there's no need to launch it.

Creating an Ontology

Design an Ontology – Understand the Domain, the Needs and the Structure

For this step I've followed the suggestions given by Natalya F. Noy and Deborah L. McGuinness in their work Ontology Development 101: A Guide to Create Your First Ontology[5]. Their paper offered a great help because they ask right questions and keep you focus on the final aim of the ontology: being a simple-to-understand and reliable source of knowledge on a certain domain.
Following a sort of top-down approach, I've gained inspiration from possible questions about history and define most general *Classes* to provide answers. From those *Classes* I've define the ontology-graph and reason about what kind of attributes (*Data-Property*) each type of node (*Class*) should had and with which kind of edges (*Object-Property*).
Once I was satisfied with the result I've moved to the implementation, but even after that I had to go back and change, remove or add many different things because, as Natalya F. Noy and Deborah L. McGuinness said in their paper:

"*Ontology development is necessarily an iterative process.*"

Also lack of experience in the field didn't help in making things right at first attempt.

Implementing an Ontology – Go with Protégé and OWL

Starting from *Protégé*'s famous Pizza Tutorial[5] and some other sites that provides examples or suggestions, with a great help from *Protégé*'s documentation[6] itself, I've been able to translate the designed structure to a working ontology with eight *classes*, eighteen *object-properties*, twenty-four *data-properties* and twenty-two *individuals*.

Ontology's Structure Analysis – Focus on Hierarchies

Before describing each class and properties, you should know some little piece of information that apply to all entities:

- Each *Class*, aside from it respective children, is disjointed with all the others, this means that an instance of a certain class will never be also an instance of another;
- *Object-properties* described below have the *Class* they are inserted in as Domain and the one inside brackets as Range;
- Every *Object-property* is Irreflexive;
- *Data-properties* associated with the type '*unsignedInt*' have the number zero if the actual value is unknown.

Protégé doesn't have an error checker, it just has a *Reasoner* which is a tool that follows rules and constraints to assume logical informations. It's a nice, but you have to activate and synchronize it manually after every update and debugging is not so clear.

Class&Properties – Character
A Character is a person that appears in the novel.

Character's Data-Properties

- *character_name(string)* → Surname and name of that person;
- *about_life(string)*→ Description of what he/she had done during his/her life;
- *age_of_death(unsignedInt)* → Self-explanatory;
- *physical_description(string)* → Self-explanatory;
- *psychological_description(string)* → Must be inferred;
- *haveTitle(unsignedInt)* → Father-property of all other achivable title, the value stands for the year of acquisition.
- *haveTitle→hasJob/isGeneral/etc...* → All sons of haveTitle inherit his type and meaning behind the value.

Character's Object-Properties

- *involved(Event)* → A character participates in a certain event;
- *involved→as_warrior/as_victim/etc…* → It's the role played by the character in a certain event;
- *relation_with(Character)* → First character interact with the second with unclear sentiment, vice versa could be different;
- *relation_with→admiration/hated/etc…* → It's the kind of relation first character has for the second. Only *married* and *relative* are *Symmetric*.

Class&Properties – Event
An Event stands for an important historical fact like a war or the foundation of a faction.

Event's Data-Properties

- *event_name(string)* → A title that describes what happened;
- *about_event(string)* → Brief description of what happened;
- *is_war(bool)* → Self-explanatory.

Event's Object-Properties

- *occurred_in(Year)* → Self-explanatory, can point to more years if the event is long enough.

Class&Properties – Location
A Location can be either a whole state or a single city.

Location's Data-Properties

- *city_name(string)* → Self-Explanatory, empty if the Location is a state;
- *state_name(string)* → Self-Explanatory.

Location's Object-Properties

- *visited_by(Character)* → If a character has been in that city/state at least once.

Class&Properties – Title
A Title is a certain recognition that a person can achieve. This *Class* can be used for represents generic or honour-related recognition, but it three children better contextualize the achievable title: a title could be a *Job*, a *MilitaryRank* or a *PoliticalOffice*. All those *Classes* share the same data and object properties.

Title's Data-Properties
- *title_name(string)* → Self-Explanatory, can stand for a group of similar jobs;
- *title_description(string)* → Brief explanation of what a certain title means and what kind of benefits and duties it grants.

Title's Object-Properties
- *achieved_by(Character)* → If a person has achieved a position, even for a couple of days, he/she will be linked to that title/job/rank/political position.

Class&Properties – Year
Romance of the Three Kingdoms covers events from the 168 to 280 AD, more or less. Each year between those two can be an instance of this Class.

Year's Data-Properties
- *year(unsignedInt)* → Self-Explanatory;
- *era(string)* → The dynasty of the Emperor define the era, but he can change it with a phrase that represents the status of the world.

Year's Object-Properties
- *was_alive(Character)* → If a person has lived in that year;

Conclusions
The results of what I've done can be seen by opening the ontology file so I'll use this section for stating my personal opinion.
Having the possibility to reason on a domain that I like using what we've learn during the natural language processing course has been a fun and interesting experience. I would like to keep enriching this ontology and maybe even expand it beyond the period of time covered by the novel. I've start developing the ontology's structure thinking on a more general historical context and in the future I'd like to discover if what I've done could work on history text book too. In that case the name of the file will definitely be *ontology-of-all-kingdoms*.

References

[1] Musen, M.A. The Protégé project: A look back and a look forward. AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 1, June 2015. DOI: 10.1145/2557001.25757003.

[2] University of Adelaide - https://www.adelaide.edu.au/

[3] Words Solver - https://words-solver.com/war-words-list/

[4] Vocabulary.com - https://www.vocabulary.com/lists/183710

[5] Ontology 101 by Natalya F. Noy and Deborah L. McGuinness - https://protege.stanford.edu/publications/ontology_development/ontology101.pdf

[6] Pizzas in 10 minutes from Protegewiki - https://protegewiki.stanford.edu/wiki/Protege4Pizzas10Minutes

[7] Protégé documentation on github - http://protegeproject.github.io/protege/views/