Relationship extraction from Sanskrit Text

Introduction:

Relationship extraction involves the detection and classification of semantic relationships mentioned within a set of artifacts, typically from text. In this project we extract family relationships from Sanskrit text by formulating it as a classification problem. We train and apply a Support Vector Machine to classify every name pair as a relation. We present a framework consisting of the following components: *person name extraction, relationship descriptor identification, and pair-wise family relationship prediction.*

CORPUS:

For the purpose of this project we use Ramayana as the corpus. The length of the text and repeating characters makes it a convincing choice for the corpus.
We scrape http://www.valmikiramayan.net/ to obtain the raw corpus.

PROCEDURE:

The relationship extraction problem is formulated as a classification problem, where each class corresponds to a relation like father, mother, sister etcetera. The relationship extraction is done through a SVM classifier. We create data points, which contain features necessary for the classifier.

PREPROCESSING:

The data points are extracted from the raw corpus using the following steps:

1) We segment the corpus based on clear delimiters such as '||', which indicate separation of shlokas in the text.
2) For each word in a segment, we extract its root form through the morphological analyzer. Due to the limitations of the tool,some words may not be processed, in which case the original form of the word is retained.
3) For each processed shloka having at least one name pair, we do the following:
    a) Identify and extract all the name pairs.
    b) For each name pair, check if a relationship exists between them using the annotated data.
    c) If a relationship exists in the above step, we create a data point for further processing.

* Please check out reference [1] if you want to know how to execute preprocessing step.

CLASSIFICATION:
There are two possible ways to approach this classification, either as a

1) Multi-class classification - Relation1 vs Relation2 vs. Relation3
Example : Father vs. Mother vs. Sister
2) Multiple binary class classification - Relation vs. Not Relation
Example : Father vs. Not Father, Mother vs. Not Mother etcetera

According to the reasearch we did on this topic, we infer that the data could be sparsely distributed for one class when compared to the other classes. This could lead to a potential issue for the multiple binary class classification approach. If there is a large amount of training data for the Not Relation class when compared to the Relation class, most of the test data will be classified as belonging to Not Relation class. Hence, we will implement the classification as a multi-class classification.

RESULTS:

We used the validation set to fine tune the various parameters of the classifier, one such parameter is TF-IDF, which scales down the impact of tokens that occur very frequently in a given corpus and that are hence less informative than features that occur in a small fraction of the training corpus. This value was empirically determined with various validation runs on the training data.

CONCLUSION:

From the results seen above we can confidently claim that approaching the relationship extraction problem using supervised learning is a feasible option. In general, extracting semantic relations between entities is an important problem in natural language processing.
Semantic relation extraction has not been extensively studied in Sanskrit, and we believe we have mentioned an elegant way to extract suitable information. The mentioned technique will sure be an encouraging factor for people interested in the topic.

Code References:
[1] : How to execute pre-processing step:
Script can be found in "*include/pre_processor.py"*
* To run this script you need to have *root_extractor.py*. root_extractor.py extracts the root form of each word using a morphological analyser given a word as input. Whereas, *root_generator.py* fetches each word from the file given as input, and outputs its corresponding root form to a new file.
* pre_processor.py has a function named "*find_relation_tuples*". That function extracts the features(words in the shloka) for a name pair if they are related.

Other important files:
* data_splitter.py : This file reads the *features.txt* entries and splits the name pairs into test and train based on the percent provided as input. Output is written to *data/split_data.txt*, this has to be read for getting the dictionary of split