Information Retrieval and Extraction

Assignment 3

Sayar Ghosh Roy (20171047)

Note that the Bengali text might not be displayed perfectly on all systems.

Introduction

My first language is Bangla or Bengali and I speak the Kolkata dialect. I have selected a collection of 15 News articles and 20 Wikipedia articles written in Bengali. These lists can be found within './URL_list/' as json files. After a manual analysis of the text content in these URLs, I was able to identify a few lexical patterns among the Named Entities (NEs). It is to be noted that these patterns simply increase the chance of an N-gram in question to be a NE, and it's not a full proof rule. Here, we try to incorporate these pattern matching based lexical rules to identify NEs of three types, namely, (1) Person, (2) Organization and (3) Place.

System

In order to visualize the performance of the rule-set on a data collection, I wrote a pipeline which collects contents from web pages given a list of URLs, tokenizes the extracted texts using a custom made tokenizer, prunes out non-Bengali information and applies the coded in lexical rules to identify named entities. The notebook for running the experiment has been provided and the code is available in this repository. The rules which rely on Part Of Speech (POS) features are not being utilized currently since that requires an external POS tagger. For collecting the contents from web pages, the 'boilerpipe3' python library has been utilized. The provided experiment code does not use multiprocessing since it is framed in a Jupyter notebook (does not allow multiprocessing) and the URL set is fairly small. To make the content population more efficient with parallel processing, consider using my content-curator pipeline.

1)

I have identified some of the following patterns in the given text, and filled in related patterns using my knowledge of the Bengali language. Almost all of these rules have exceptions in the real world. However, similar patterns can be seen in the existing literature where lexical features and rules have been utilized for performing Named Entity Recognition in Bengali. This system description paper from LTRC, IIIT Hyderabad uses a collection of similar patterns. The following patterns within tokens will allow us to mark it as a starting (or ending) point for a particular type of Named Entity.

Person:

1. The token is an honorific indicating the start of a name such as 'শ্রীমতি', 'শ্রী', 'শ্রীযুক্ত', 'মিঃ', 'মিস', 'মিসেস', 'মিস্টার', 'মাস্টার'
2. The token is a profession identity marker such as 'ডঃ', 'ডাক্তার', 'ডক্টর', 'স্বামী', 'সৈয়দ', 'রেভারেন্ড', 'প্রফ', 'অধ্যাপক', 'প্রধানমন্ত্রীর'
3. The token is an honorific marking the end a name such as 'বেগম', 'বিবি'
4. The token ends with a relation marker such as 'বাবু', 'দাদা', 'দা', 'সাহেব', 'কাকু'
5. The token ends with an indicator of plurality such as 'রা', 'এরা', 'দের'
6. The token ends with an indicator of reference or identity marker such as 'কে' provided the token is not 'থেকে'
7. The name contains the marker ' ঁ' indicating that the person has expired
8. The token is a known last name such as 'ঘোষ', 'বোস', 'বসু'
9. The token is a known middle name such as 'কুমার', 'চন্দ্র'
10. Rule out tokens which contain the character 'ৎ' as it rarely occurs in proper nouns

Place:

1. The token indicates the end of a street name such as 'সরনি', 'রোড', 'স্ট্রিট', 'লেন',
2. The token indicates the end of an educational institution name such as 'স্কুল', 'বিদ্যালয়', 'কলেজ', 'বিশ্ববিদ্যালয়', 'মহাবিদ্যালয়'
3. The token indicates the end of an identifier for a waterbody such as 'নদী', 'লেক', 'হ্রদ', 'সাগর', 'মহাসাগর', 'জলা', 'ঝিল', 'খাল', 'বিল', 'পুকুর'
4. The token indicates the end of the name of a hill or a mountain such as 'পাহাড়', 'পর্বত', 'টিলা'
5. The token ends with an area type marker such as 'নগর', 'গঞ্জ', 'গ্রাম', 'পুর', 'গড়'
6. The token indicates the end of a place for receiving medical treatment such as 'হাসপাতাল', 'ক্লিনিক', 'স্বাস্থ্যকেন্দ্র', 'আরোগ্যকেন্দ্র', 'নিরাময়কেন্দ্র'
7. The token indicates the name of a village such as 'গ্রাম', 'গাঁ'
8. The token 'থানা' indicating the end of a police station name
9. The token indicates the name of a garden such as 'বাগ', 'বাগান', 'বাগিচা'
10. Two tokens ('নার্সিং', 'হোম') to end the name of a nursing home

Organization:

1. An N-gram of size greater than 2 contains only tokens of 1 or 2 characters indicating an acronym
2. A token from the set ('সনস', 'সন', 'ডটার', 'ডটারস', 'ডটার্স') follows the word 'এণ্ড' indicating the name of a shop or a commercial establishment
3. A token from the set ('কম্পানি', 'কোম্পানি') follows the word 'এণ্ড' indicating the name of a corporate merger
4. The token indicates the end of a manufacturing company's name such as 'এন্টারপ্রাইস', 'কোম্পানি', 'কোম্পানি'
5. The token indicates the name of a people's union such as 'সমিতি', 'সমাজ', 'কোয়াপারেটিভ', 'কমিটি'
6. The token indicates the name of a laboratory such as 'ল্যাব', 'ল্যাবরেটরি', 'রসায়নাগার', 'গবেষণাগার'

7. The 'কর্পোরেশন' indicating the end of a corporation's name
8. The token 'পরিষদ' indicating the end of a managing committee or government branch identifier
9. The token 'ইনস্টিটিউট' indicating the end of any generic institution's name
10. The token pairs ('পাবলিক', 'লিমিটেড') or ('প্রাইভেট', 'লিমিটেড') indicating the end of a company name

For examples of the above rules in use, refer to the provided tagged data. We combine the above lexical rules with POS tag information to identify the correct boundaries. Using NNP and NNPS tags, we can identify the number of tokens following (or preceding or surrounding - as the case may be) a particular lexical anchor pattern. Say 'Dr. Mike Tyson said that...' exists, and we identify 'Dr.' as the lexical anchor using rule 2 for Person identification, then we can easily identify 'Dr. Mike Tyson' as the named entity since Mike and Tyson will receive NNP tags from the POS tagger. In the absence of a treebank or a reliable POS tagger, we simply use heuristics based on observed frequencies to achieve the boundary identification. For example, in case of Proper Names, the usual size is of two tokens, the name of a garden usually has one token, and so on.

2)

The rules above were based on the contents of the selected web pages and therefore, they perform as intended. The main issue in practice is due to the improper boundary identification which can be mitigated by the use of a POS tagger as described above. Apart from that, the rules pick up certain pieces and wrongly classify them as named entities. The exception handling for every rule leads to cascading effects: more conditions and more exceptions. Consider: Person Rule 6 which contains one unit of pattern to handle a popular exception. It incorrectly produces (486, 'দিকে', 'per') [format: (index, token, label)] which is the marker of a direction (not a NE) as a person and it also produces (429, 'কলকাতাকে', 'per') which should have been classified as a location 'loc'. We also have failures like 'মর্যাদা' which is an adjective but is being classified as person name as it ends in the marker 'দা' (used to refer to a senior male individual). Still, to appreciate the rules, we have instances such as {(805, 'ভারত', 'org') (806, 'কোম্পানি', 'org')} referring to the East India Company, (1333, 'নেপোলিয়নকে', 'per') identifying Napoleon and {(979, 'প্রনয়নকারী', 'org') (980, 'পরিষদ', 'org')} referring to a committee which procures materials. Sometimes, the rule identifies the NE correctly but does not have the granularity to identify the correct NE type. Any rule based system follows the law of diminishing returns. Thus, the amount of exceptions and cases which can be handled and the system performance increase per rule, keeps decreasing exponentially with increase in number of rules. Now, with 30 rules in total, expecting great results is naive. However, if the lexical rules were combined with the POS features, we could handle many more cases.

Note that the entire analysis is based on the produced outputs of the actual implementations of the rules. Thus, the analysis is free from possible interpretations of various conditions and follows a uniform application of a particular rule throughout. For more details, refer to the provided notebook. The raw results can be viewed in './raw_results'.

3)

A major issue faced above is in determining the order of application of rules i.e which rule should take precedence in a scenario where multiple rules are applicable. Thus, instead of specifying if-then conditions for timely applications, a natural next step is to consider features and look at the problem as a sequence labelling task. Thus, Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) operating on the principle of Maximum Likelihood Estimation (MLE) come in handy here. HMMs are generative in nature and maximizes the joint probability of the sequence of labels and the input text sequence. CRFs subtly differ from HMMs since they maximize the conditional probability of the label sequence given the input text. Thus, CRF is a discriminative setting. HMMs and CRFs are both supervised settings and we would typically have our training text for, labelled with BIO tags. The HMM or CRF (or a seq2seq model using RNNs or Transformers) would utilize its learning algorithm (Baum-Welch for HMMs, lbfgs solver for CRFs) to tune itself such that the probability (joint probability or conditional probability as the case may be) of labels given text plus feature sequence is maximized. A semi-Markov CRF approach combines the best of both worlds and has been shown to improve[1] NER performance over simple CRFs in certain cases.

There are a set of advantages in moving from a purely rule based setting to a sequence labelling setting. Firstly, the implicit use of probabilities come into the picture. Now, we have more granularity as opposed to binary applications of hardcoded rules. Secondly, various features can be included as part of the input. Apart from just the lexical token, we can have the POS, morph features, labels such as GNP (Gender Number Person) and even semantic embeddings. Lastly, we can combine hard constraints based on a dictionary or labelled N-grams with the Machine Learning (ML) model by simply introducing those as binary features such as 'is a known Multi Word Expression (MWE) or not'. The possibilities of trying out various experiments by swapping out feature sets is endless for these types of sequence labelling models. For example, for morphologically rich languages, using chunk information, root word, affixes collection, and labels such as 'is it the start of a sentence?' has been experimentally found to be beneficial.[2]

There are other questions regarding the input sequence itself and ordering of Natural Language Processing (NLP) tasks to be answered here and some design choices to be made. Should we use a morph splitter first and have a sequence composed of root words and affixes, or should we include {root: <>, affixes: <()>} as a single unit? In which order should we perform the tasks of chunking, NER and MWE recognition? There are quite a few possibilities here which should be kept in mind but clearly, the issues witnessed in the labelling[3] can be overcome to a great extent by shifting to a ML approach. The only drawback here is the requirement of supervised data. Labelling is often laborious, time consuming and expensive (in the sense that it requires trained annotators).

4)

For Indian Languages (ILs), Conditional Random Fields (CRFs) are preferred over Hidden Markov Models (HMMs) primarily because the Markovian assumptions do not always hold true. ILs such as Hindi, Bengali, Marathi, Oriya, etc have free word order which makes sequence labelling and long

[1] Refer here
[2] Refer here
[3] Refer to provided results if interested

range dependency modeling difficult. For example, in Hindi: ‘raam ne khaana khaaya’, ‘khaana khaaya raam ne’, ‘khaana raam ne khaaya’, ‘khaaya raam ne khaana’, ‘khaaya khaana raam ne’ (ram eat food, food eat ram, food ram eat, eat ram food, eat food ram) are all valid (pragmatics might differ, but semantics remain the same). Hence, these issues make things difficult to model as a sequence labelling task. ILs do not have markers such as first word capitalization for marking proper nouns. Foreign words, borrowed words, and the spelling norms for nativisation are non-standardized. These cause problems while formulating patterns as the same sounding name may be spelt differently by different annotators. Modeling Dravidian languages such as Telugu, Malayalam, Tamil, Kannada, etc involve rich and robust morph analyzers as the words themselves are full of inflectional and derivational affixes. Indo Aryan Languages have words formed by *Sandhi* and sandhi-splitting is a common pre-processing step while handling data from these languages. Almost all ILs have agglutination and defining a universal data structure for storing agglutinative markers is an involved task. Also, adapting sequence labelling models for ILs is a challenging task due to the amount of available possibilities for feature sets as seen in part 3. Last but not the least, ILs are resource poor and finding good amounts of labelled data especially for Indo-Tibetan languages like Nepali is particularly tricky.