# INLP 2015-2016 Final exam.

Here you can find a possible solution of the exam

We are interested on automatically building a gazetteer of demonyms for allowing associating a country or city to their demonyms, i.e. the name given to their inhabitants (France → French, Peru → Peruvian, and so). We will use the Wikipedia as knowledge source.

Searching the English WP by "demonym" the following information (summarized) is obtained. In the retrieved WP page 500 demonyms occur. A sample of relevant lines (a lot of noise occurs in the page) follows:

| | | |
|---|---|---|
| Africa | → | African |
| Italy | → | Italian |
| Poland | → | Polish |
| Croatia | → | Croatian (also "Croat"), |
| (North / South] Korea | → | [North / South] Korean, |
| Hanoi (Vietnam) | → | Hanoian |
| Iran | → | Iranian (also "Irani" or "Persian") |
| Florence | → | Florentine (also Latin "Florentia") |
| Ann Arbor | → | Ann Arborite |
| Israel | → | Israelite (also "Israeli", depending on the usage; see below) |
| Netherlands | → | Netherlander (though see below; Irregular forms) |

You should answer the following questions:

1. (3 points) Propose a way of using FS technology for facing the problem of getting from a location (specifically from a country, a city, or a state, province, etc.) the possible set of valid demonyms (note that LOCATION → DEMONYM can be seen as a morphological derivation). Sketch a FS architecture of the system (FSA or FST). Define accurately the components of the system. Illustrate them with some examples, from the ones included above, or proposed by yourselves. Use freely the FS operations you need (concatenation, intersection, union, etc.). What we want if to face the problem of more or less regular demonymy, i.e. from "Italy" to "Italian", not the irregular cases as "Iran" to "Persian, which will produce overfitting.

**The first task should be collecting from the "demonym" page of EWP as many pairs <location, demonym> as possible.**

A simple way of carrying of this task can be writing a set of regular expressions for extracting such pairs.

For instance, for extracting the pair <Virginia, Virginian> from the line:

```
        [[Virginia]]  → Virginian
```

the RE:

```
        u'^\*\[\[([\w]+)\]\] \u2192 ([\w]+)$'
```

can be used (the Unicode of → is  \u2192).

For  extracting the pairs  <Croatia, Croatian> and <Croatia, Croat> from the line:

```
        *[[Croatia]] \u2192 Croatian (also "\'\'Croat\'\'")
```

the RE could be:

```
        u'^\*\[?\[?([\w]+)\]?\]? \u2192 ([\w]+) \(also [^\w]*([\w]+)
[^\w]*\)$'
```

Looking at the "demonym" page of EWP we can see that about 10 rules similar to these two could be enough for extracting the most productive training pairs.

Using a python script we can collect 743 lines from  the "demonym" page of EWP.  Applying the first RE we obtained 349  pairs, while applying the second rule 37 pairs are obtained. We hope that writing about 10 rules more than 500 pairs could be extracted.

The second step will be building a set of FST mapping locations to demonyms. Consider the following pairs obtained using the procedure in first step (all of them obtained through the application of the first rule):

```
        <u'Oamaru',     u'Oamaruvian'>
        <u'Oslo',       u'Oslovian'>
        <u'Peru',       u'Peruvian'>
        <u'Warsaw',     u'Varsovian'>
        <u'Niger',      u'Nigerien'>
```

For knowing how many and which FST is needed we will decompose each of the pairs into two parts, the common stem and the set of

**possible suffixes. For instance, for the pair <u'Peru', u'Peruvian'> the following possibilities are allowed (where for each line the first item is the stem, the second the location suffix, and the third the demonym suffix:**

| | | |
|---|---|---|
| **'Peru'** | **''** | **'vian'** |
| **'Per'** | **'u'** | **'uvian'** |
| **'Pe'** | **'ru'** | **'ruvian'** |
| **'P'** | **'eru'** | **'eruvian'** |

**The same is performed for all the pairs. For instance for <(u'Oslo', u'Oslovian'>:**

| | | |
|---|---|---|
| **'Oslo'** | **''** | **'vian'** |
| **'Osl'** | **'o'** | **'ovian'** |
| **'Os'** | **'lo'** | **'lovian'** |
| **'O'** | **'slo'** | **'slovian'** |

**For each pair of suffixes (as $p_1$ = <'','vian'> or $p_2$ = <'o','ovian'>), that we will name a "suffix_pattern_pair", and refer to the first element as the "location pattern" and to the second as the "demonym pattern", we compute:**

- **The coverage, i.e. how many pairs in the training data matches this pattern, in our example how many pairs <location, demonym> matches the suffixes '' for the location and 'vian' for the demonym, with a common stem. The two examples, <u'Peru', u'Peruvian'> and <(u'Oslo', u'Oslovian'> matches but also many others as <u'Oamaru', u'Oamaruvian'> above. The last two examples above (<u'Warsaw', u'Varsovian'> and <u'Niger', u'Nigerien'>) do not match. In this way we compute for each pair of suffixes p the coverage(p).**

- **The accuracy, i.e. What is the ratio of correct matches against all matches of the location part of the pattern. For instance for $p_1$, as the location pattern is '' every pair matches, while for $p_2$ as the location pattern is 'o' only the locations ending with 'o' match. In this way we compute for each pair of suffixes p the accuracy(p).**

**It is easy to see that for a small length of the location pattern (zero, one characters) the coverage is very high but the accuracy very low. For long location patterns (>3 characters) the accuracy is high but the coverage small . So we must examine the distribution of these two measures for getting a good balance.**

**For instance for the pattern:**

**<'u','uvian'>**

**only two pairs are found on the dataset:**

**<'Oamaru', 'Oamaruvian'>**

**and**

**<'Peru', 'Peruvian'>**

**As there are 386 pairs extracted through our two rules**

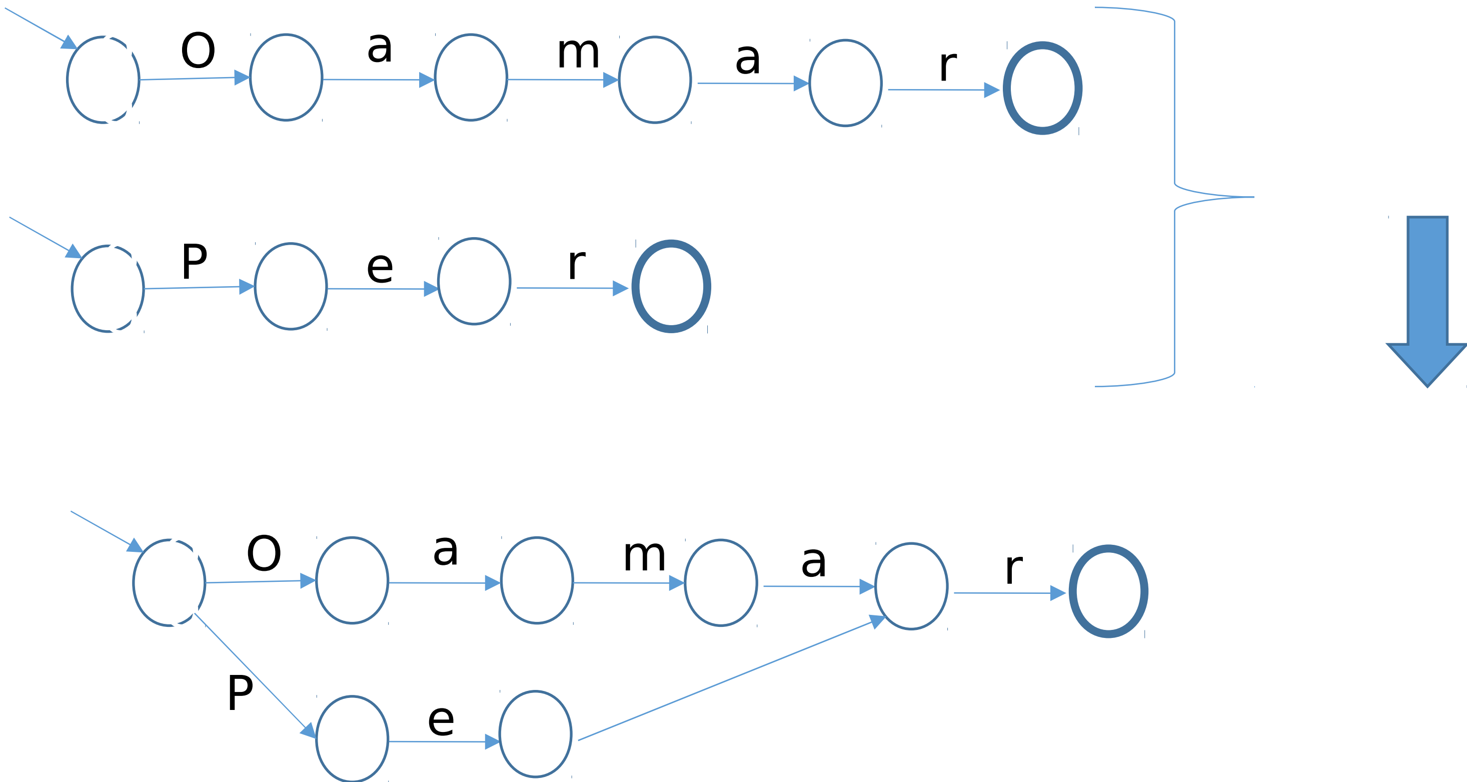
**coverage(<'u','uvian'>) = 2/386 = 0.0052**

**There are 7 locations in our dataset ending with 'u' so:**

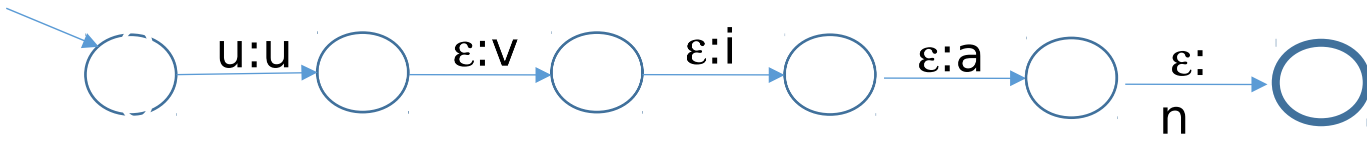
**accuracy(<'u','uvian'>) = 2/7 = 0.285**

**We can build our recognizer as a FST consisting of the concatenation of a $\text{FSA}_{\text{stem}}$ and a $\text{FST}_{\text{suffix.}}$ The $\text{FSA}_{\text{stem}}$ will recognize/generate the stem while the $\text{FST}_{\text{suffix.}}$ will recognize/generate the suffixes**

**Building $\text{FST}_{\text{suffix.}}$ is straightforward. We just have to assign a FSA to each stem ('Oamar' and 'Per') and build a FSA as the union of both.**



**Building the a FST$_{suffix.}$ is not especially difficult. We consider the location pattern as the upper level and the demonym pattern as the lower level. We can, thus recognize a location producing the demonym or recognizing the demonym producing the location (or recognizing the pair or generating the pair).**



2. (3 points) Propose a way of building automatically as many of possible basic components of the system (roots, affixes, phonological changes, morphotactic)

**We have built the FSA for recognizing/generating the stems automatically from the "demonyms" EWP article. The problem, obviously, is that only the stems occurring in this page are recognized. An alternative is using as FST$_{stem.}$ an universal acceptor (the FSA equivalent to the RE ".*"). Another not so loose acceptor is discussed in next section.**

**FST$_{suffix.}$ Has been automatically learned from the same WP page, in this case with the chosen accuracy/coverage ratio.**

**In this way a set of several FSA+FST are built one for each "suffix_pattern_pair".**

**No phonological changes have been taken into account in our approach (they are included within the FST).**

**The morphotactic is simply implemented as associating different FST$_{stem}$ and FST$_{suffix}$ pairs, i.e. for each FST$_{suffix}$ a FST$_{stem}$ is used.**

3. (2 points) Building a FST for performing LOCATION $\rightarrow$ DEMONYM derivation, e.g. locating "Italian" in a text knowing that it refers to a person from "Italy" is not difficult. The main problem, however, is learning the roots and associated morphotactics, e.g. from Poland we derive Polish but from Ireland we do not derive Irelish but Irish. For building a gazetteer this is needed. These idiosyncratic constraints cannot be captured from a set of only 500 examples. Using Wikipedia for providing a greater set of training examples can be a solution to this issue. Propose an approach to face this problem using WP pages (which ones?). The idea is that the WP pages of Poland and Ireland, for the example above, could provide useful information for deriving new rules or constraining existing ones.

**Consider the pattern p = <'u','uvian'> discussed above. If the FSA is the one included above only the two stems 'Per' and 'Oamar' can be recognized, if the universal acceptor is used whatever stem can be recognized. Obviously the first option is correct but almost useless because it covers only 2 stems, while the second is too loose (the pattern could be applied to whatever word ending with 'u'). For instance 'Corfu' could be mapped applying the pattern p into the incorrect demonym 'Corfuvian' while the correct demonym is, according to the WP page, 'Corfiot'.**

**One possibility for selecting the correct stems (i.e. $FST_{stem}$) for each pattern p is the following:**

**We get from EWP the set of all populated locations (countries, cities, political administrations, provinces, states, etc.). The set can be easily collected using as anchors the corresponding categories and subcategories in EWP and through them the corresponding pages. Links category $\rightarrow$ subcategory and category $\rightarrow$ page can be used for it. For each location name we apply the set of patterns obtaining a set of demonym candidates. It is likely that only the correct ones occur in the text of the EWP article. For instance, for 'Corfu' the candidates include 'Corfuvian' (if p is applied) together with other possible terms (perhaps 'Corfuan', 'Corfish', 'Corfate', and, likely, the correct one, 'Corfiot'. We look within the page for all these terms and their morphological variants) and if only one is found we include the stem of the term within the corresponding $FST_{stem}$ . In our case we have been lucky and different variants of 'Corfiot' occurred in the page while none of the other candidate occurred. Processing the page and looking for the set of demonym candidates is not difficult.**

4. (2 points) Which NLP tools (processors) should be used for the previous

task?

**As can be easily seen the NLP tools needed are rather simple. The first step of our approach uses simply RE matching on the EWP “demonym” page. The page could be accessed and managed using python modules (I have used wiki tools). Building the FSA and FST can be done also without problems using whatever FS tool.**

**Regarding the 3rd step, EWP can be accessed also with whatever WP python tool (I have used too, wiki tools). The process of the page is simple. Some cleaning of the EWP page can be performed. The only NLP tool needed is a lemmatizer. WordNet can provide one but it is limited to WordNet coverage and, so, not all the demonyms exist ('Corfiot', for instance does not occurs in WordNet). You can use, instead a morphological analyser (Freeling, Stanford).**