Named Entity Matching Method Based on the Context-free Morphological Generator

Jan Kocoń and Maciej Piasecki

{jan.kocon, maciej.piasecki}@pwr.edu.pl
Institute of Informatics
Wrocław University of Technology
Wybrzeże Wyspiańskiego 27,
Wrocław, Poland

September 17, 2014

Politechnika
Wrocławska

.

Utilisation of Resources

Lexicons of proper names (PNs)

Valuable resources for many natural language processing (NLP) tasks, i.e. Named Entity Recognition (NER). Mostly contain basic morphological forms (lemmas) of PNs. Most PN inflected forms in text cannot be straightforwardly matched in lexicon.

- inflected PN: $[\text{Lidze}_{\text{lemma=0}}\ \text{Polskich}_{\text{lemma=0}}\ \text{Rodzin}_{\text{lemma=0}}]_{\text{PNlemma=0}}$
- lemma: $[\text{Liga}_{\text{lemma=1}}\ \text{Polskich}_{\text{lemma=0}}\ \text{Rodzin}_{\text{lemma=0}}]_{\text{PNlemma=1}}$
- lemmatiser: $[\text{Liga}_{\text{lemma=1}}\ \text{Polski}_{\text{lemma=1}}\ \text{Rodzina}_{\text{lemma=1}}]_{\text{PNlemma=0}}$

Issues

Jesteśmy już w **Warszawie**.
(eng. *We are already in Warsaw*)

Jesteśmy już w **Warzsawie**.
(eng. *We are alreay in Warsaw*)

Miasteczko **Werszawa** ma nazwę bardzo podobną do nazwy stolicy Polski.
(eng. *Werszawa town has a name very similar to the name of the Polish capital*)

On już dawno opuścił **Zieloną Górę**.
(eng. *He left Zielona Góra a long ago.*)

Issues

Summary

- PN lexicons are large.
- PNs are mostly unknown words (or Out of Vocabulary Words).
- Polish is a language with rich inflection.
- Complex inflection of the Multi-word PNs.

Goal

- Recognition of unknown word forms in Polish texts.
- Special focus is given to the unknown inflected PN forms that are included in a large PNs lexicon.
- Efficient matching of words in PNs lexicon.

Related Works

- Simple string similarity metrics, e.g. Overlap Coefficient, Soundex, Levensthein, Common Prefix δ (CP_δ), others; e.g.:

$$\mathrm{CP}_\delta(s,t) = \frac{(|\,\mathrm{lcp}(s,t)| + \delta)^2}{|s| \cdot |t|}$$

  - $\mathrm{lcp}(s,t)$ – the longest common prefix of given strings: s & t;
  - δ – equals 1 if one of the two given strings ends with a, and the second ends with: *o, y, ą, ę*, else 0.

- Combine similarity metrics into a complex one (classifier-based approach using Logistic Regression) – Named Entities Similarity Function (NamEnSim).

Idea

- Match any pair of words which share the same lemma.
- Store entries in lexicons in their proper forms.
- Group **similar** entries in morphological dictionary.
- Create inflection rules by processing the groups.
- Generate all possible word forms of any unknown text word.
- Match the result against the known set of constituents from gazetteers (e.g. PNs constituents).

Example of Group

sprawom	sprawa	subst:pl:dat:f*
sprawie	sprawa	subst:sg:loc:f
sprawą	sprawa	subst:sg:inst:f
sprawie	sprawa	subst:sg:dat:f
sprawę	sprawa	subst:sg:acc:f
sprawo	sprawa	subst:sg:voc:f
sprawy	sprawa	subst:pl:voc:f
sprawy	sprawa	subst:pl:nom:f
sprawy	sprawa	subst:pl:acc:f
sprawami	sprawa	subst:pl:inst:f
sprawach	sprawa	subst:pl:loc:f
spraw	sprawa	subst:pl:gen:f
sprawa	sprawa	subst:sg:nom:f
sprawy	sprawa	subst:sg:gen:f

*in the given example tags come from the National Corpus of Polish Tagset and denote the following attributes (separated by the colon):

grammatical category : number : case : gender

Endings

$K_G = \{$'om','a','ie','ą','ę','o','y','ami','ach',ø$\}$
$C^2_{K_G} = \{$
{'om','a'},{'om','ie'}, {'om','ą'}, ...,
{'om',ø}, {'a','ie'}, {'a','ą'}, ...,
{'a',ø}, ..., {'ach',ø}
$\}$
$B_S = \{$
{{'om','a'}, 1},{{'om','ie'}, 2}, {{'om','ą'}, 1}, ..., {{'om','y'}, 4}, ...,
{{'om',ø}, 1}, {{'a','ie'}, 2}, {{'a','ą'}, 1}, ...,{{'a','y'}, 4}, ...,
{{'a',ø}, 1}, ..., {{'ach',ø}, 1}
$\}$

Inflected Forms Generator

In the following, let:

- $s =$ 'Warszawie'
- $B_S =$ { {{'ie','a'}, 4},{{'om','ie'}, 2}, {{'om','ą'}, 1}, {{'e',ø}, 1}}

1. Select a subset BS_S of the set B_S of such elements, in which at least one of the endings is the ending k of word s.
2. For each element BS_S such as $\{\{k, ks\}, a\}$ (e.g. $BS_S =$ { {{'ie','a'}, 4}, {{'om','ie'}, 2}, {{'e',ø}, 1} }):
   1. Create a word gs by removing the ending k from the word s and adding the ending ks
   2. Add the pair $\{gs, a\}$ to the set W, e.g. $W =$ {{'Warszawa', 4}, {'Warszawom', 2}, {'Warszawi', 1}, {'Warszawiee', 1}}
3. Return the set W

Generator as Similarity Function

In the following, let:

- $s =$ 'Warszawie'
- $N =$ {'Warszawa', 'Kraków', 'Wrocław', 'Werszawa', 'Warszawom'} – a proper names dictionary as a set of strings

$W =$ {{'Warszawa', 4}, {'Warszawom', 2}, {'Warszawi', 1}, {'Warszawiee', 1}}

The result is a subset of $\{gs, a\}$ from W where $gs \in N$. In the following example the result is { {'Warszawa', 4}, {'Warszawom', 2} }

Test Set

For the purposes of the tests, we used *NELexicon* – a very large lexicon of about 1.4 million Polish PNs and NEs available on the Creative Commons licence. It includes not only lemmas, but also inflected word forms for some PNs.

1. a *small search space (0 mode)* – only base forms of the test examples,
2. a *large search space (1 mode)* – all base forms from the named entity category.

Category	Size		
	(s_space)	(l_space)	Tests
person_first_nam	480	15208	1720
country_nam	157	332	621
city_nam	8144	38256	30323

Table : 1. Test sets and search spaces for different experiment modes and categories

Quality Measures

- a – the number of all test examples,
- s – the number of tests, in which a single result was returned,
- m – the number of tests with more than one result returned,
- sc (*single correct*) – the number of tests, in which a single result was returned and it was correct,
- mc (*multiple and correct*) – the number of tests with more than one result, but including the correct one,
- $mc2$ (*multiple with best one correct*) – the number of tests with more than one result and with the correct result as the top one (i.e. having the highest decision function value).

We used three measures:

- *All answer accuracy*: $\mathrm{AA} = \frac{sc}{s+m}$
- *Single result accuracy*: $\mathrm{SR} = \frac{sc}{s}$
- *Relaxed all answer accuracy*: $\mathrm{RAA} = \frac{sc+mc}{s+m}$

We propose modified measures:

- *Modified all answer accuracy*: $\mathrm{mAA} = \frac{sc+mc2}{s+m}$
- *Modified global accuracy*: $\mathrm{mACC} = \frac{sc+mc2}{a}$

Results

similarity metric	resource	mAA	SR	RAA	mACC
NamEnSim5	person_first_nam0	0.9862	0.9946	0.9952	0.9564
	person_first_nam1	0.8905	0.9908	0.9875	0.8703
	country_nam0	**0.9967**	**1.0000**	**1.0000**	0.9614
	country_nam1	0.9883	0.9980	0.9983	**0.9549**
	city_nam0	0.9412	0.9884	0.9883	0.9132
	city_nam1	0.8260	0.9765	0.9740	0.8132
CP_δ	person_first_nam0	0.9683	0.9810	0.9812	0.9593
	person_first_nam1	0.7915	0.8636	0.8662	0.7878
	country_nam0	0.9885	0.9950	0.9951	**0.9678**
	country_nam1	0.9672	0.9866	0.9869	0.9501
	city_nam0	0.9175	0.9322	0.9306	0.9168
	city_nam1	0.7734	0.8168	0.8170	0.7733
RuleSim	person_first_nam0	**0.9924**	**0.9981**	**0.9982**	**0.9826**
	person_first_nam1	**0.9366**	**1.0000**	**0.9982**	**0.9273**
	country_nam0	0.9950	**1.0000**	**1.0000**	0.9533
	country_nam1	**0.9899**	**1.0000**	**1.0000**	0.9485
	city_nam0	**0.9880**	**0.9998**	**0.9998**	**0.9328**
	city_nam1	**0.8887**	**0.9984**	**0.9986**	**0.8401**

Table : 2. Evaluation results for the method utilising a morphological generator (RuleSim) in comparison with CP_δ and NamEnSim5 – small search space (0 mode), large search space (1 mode)

Summary

+ It is possible to obtain reasonable results without using complete morphological dictionary (PN matching task).
+ Achieved results are better than results of NamEnSim5 complex similarity function based on Logistic Regression.
+ RuleSim Similarity function value can be used as a feature in more complex NER methods.
+ The proposed method should achieve similar results for languages where the suffix plays the major role in word inflection.
- The proposed method is highly dependent on the quality of the PN dictionary.
- The proposed method does not solve the problem of the misspelled PNs in both dictionary and text.

The End

Thank you for your attention.

Baseline

Similarity metric	s_space variant			l_space variant		
	AA	SR	RAA	AA	SR	RAA
ChapmanLengthDeviation	0.32260	0.57387	0.53503	0.06	0.30721	0.40293
Jaro	0.83164	0.86895	0.87062	0.30501	0.64447	0.63590
JaroWinkler	0.84859	0.87275	0.87514	0.55599	0.64407	0.66517
MatchingCoefficient	0.74011	0.96608	0.97119	0.32133	0.93148	0.94260
Soundex	0.66158	0.97502	0.97401	0.63084	0.68395	0.69893
OverlapCoefficient	0.76780	0.82815	0.83164	0.61171	0.67016	0.68149
QGramsDistance	0.85198	0.86717	0.86893	0.61902	0.67568	0.68542

Table : 3. Baseline test for person_first_nam with small (s_space) and large (l_space) search space