A Memory-Based Tagger for Polish

Adam Radziszewski
Tomasz Śniatowski

Institute of Informatics
Wrocław University of Technology

Outline

- Tagging Polish
- Memory-Based Tagging
- Proposed algorithm
- Implementation
- Evaluation
- Simple majority voting
- Conclusions

Tagging Polish: background

- POS tagging: assigning morpho-syntactic tags to tokens
- Rich inflection → tags specify Parts of Speech + inflectional properties, such as grammatical case, verb aspect
- MSD (Morpho-Syntactic Description) tagging
- Two large manually annotated corpora for Polish:
  - IPI PAN Corpus: 880,000-token manually annotated part
  - National Corp. of Polish: 1220,000-token manually ann. part
  - Both have similar tagsets

Tagging Polish: tagsets

- Each grammatical class (~POS) designates a set of attributes (gram. categories) whose values must be given
- E.g. nouns are specified for number, gender and case
- Over 30 classes and over 10 attributes in both tagsets
- Over a thousand different tags appearing

subst:sg:nom:f — a singular, nominative, feminine **noun**

praet:pl:f:imperf — a plural, feminine, imperfective **past verb**

Tagging Polish (3)

- Two popular taggers made specifically for Polish
  - TaKIPI (Wrocław Univ. of Technology)
    - Small set of hand-written rules
    - Substantial number of decision trees to acquire rules from ref. corp.
    - Hard-coded to the IPI PAN Corpus tagset
  - PANTERA (Inst. of Comp. Sci, Polish Academy of Sciences)
    - Modified version of Brill's transformation-based learning
    - Configurable tagset
- Both assume two-stage operation: morph. analysis and contextual disambiguation (tag elimination)
- Both employ tiered tagging (tag parts dealt with separately)

Memory-Based Tagging

- Memory-Based Learning (MBL): storing training examples without generalisation
- Classification of a new instance:
  - *k* nearest neighbours retrieved using a given similarity metric
  - majority voting used to get the class label (possibly with a distance weighting scheme)
- Memory-Based Tagging:
  - Context of each token represented as fixed-width feature vector (e.g. surrounding wordforms, ambiguity classes)
  - A popular module called MBT used directly for Polish gave unacceptable results (86,9% Weak Correctness)

Proposed algorithm (1)

- Idea: benefit from the positional character of the tagset
- Training results in a separate case base for grammatical class and each tagset attribute (tiered tagging)
- Morphological analysis first, we deal only with disambiguation here
- Parametrised with a set of features for each attribute
  - Feature — function *token context* → set of *string*/*symbol* values
  - E.g. possible values of grammatical case of the token preceding the one being disambiguated (position = *-1*)
- Parameters of the MBL algorithm (*k*, sim. metric, weighting)

Proposed algorithm (2)

TRAIN (sentence)

- For attr in [class, attr_1, attr_2, …, attr_k]:
  - For each token in sentence:
    - if token ambiguous w.r.t attr:
      - **GEN EXAMPLE** for token into case_base(attr): **<**features(attr), correct value of attr**>**
      - remove tags from token with incorrect values of attr

Example:

class has been dealt with

- training examples generated
- incorrect values removed

now attr = *number* (values: **sg** or **pl**)

1

Boję

fin : **sg**:pri:imperf
~~subst~~:sg:acc:f

number(tok1) = {**sg**}
No ambiguity here

2

się

qub

number(tok2) = {}
No ambiguity here

3

myszy *(I'm afraid of mice)*

subst:**sg**:gen:f
subst:**sg**:dat:f
subst:**sg**:loc:f
subst:**sg**:voc:f
subst:**pl**:nom:f
subst:**pl**:gen:f ← gold standard
subst:**pl**:acc:f
subst:**pl**:voc:f

number(tok3) = {**sg**, **pl**}
Ambiguous!

Example generation (*current pos* = tok3):

```
class[-1] = {qub}   # tok2
class[0]  = {subst} # tok3
class[+1] = {} # out of sent
number[-1]= {}      # tok2
number[0] = {sg,pl} # tok3
number[+1]= {} # out of sent
Correct class label = pl
```

Leave only tags: *number*(tok3) = **pl**

subst:pl:nom:f
subst:pl:gen:f
subst:pl:acc:f
subst:pl:voc:f

Then attr = *case* (values: **nom**, **gen**, **acc**, …)

10

Example:

number has been dealt with

- training examples generated,
- incorrect values removed

now attr = *case* (values: **nom**, **gen**, **dat**, **acc**, **loc**, **inst**, **voc**)

1

Boję

fin : **sg**:pri:imperf
~~subst~~:sg:acc:f

case(tok1) = {}
No ambiguity here

2

się

qub

case(tok2) = {}
No ambiguity here

3

myszy *(I'm afraid of mice)*

subst:~~sg~~:gen:f
subst:~~sg~~:dat:f
subst:~~sg~~:loc:f
subst:~~sg~~:voc:f
subst:pl:**nom**:f
subst:pl:**gen**:f ← gold standard
subst:pl:**acc**:f
subst:pl:**voc**:f

case(tok3)={**nom**,**gen**,**acc**,**voc**}
Ambiguous!

Example generation (*current pos* = tok3):

```
class[-1] = {qub}   # tok2
class[0]  = {subst} # tok3
class[+1] = {} # out of sent
number[-1]= {}      # tok2
number[0] = {pl} # tok3
number[+1]= {} # out of sent
Correct class label = pl
```

Leave only tags: *case*(tok3) = **gen**
subst:pl:**gen**:f

Then attr = …

Proposed algorithm (3)

DISAMBIGUATE (sentence)

- For attr in [class, attr_1, attr_2, …, attr_k]:
  - For each token in sentence:
    - if token ambiguous w.r.t attr:
      - wanted_val = CLASSIFY(token, sentence, case_base(attr))
      - if wanted_val in values of attr in token:
        - remove tags from token with other values of attr
- For each token in sentence:
  - Force one tag per in token if multiple left (prefer 'tagset-first')

Implementation

- WMBT: Wrocław Memory-Based Tagger (GPL'ed)
- 421 lines of Python code, using:
  - TiMBL (MBL classifier, Tilburg University)
  - WCCL (feature extraction formalism & toolkit)
- Features currently employed
  - Class, number, gender and case in window (-3, -2, -1, 0, 1, 2)
  - Wordforms in the window if frequent in training data (500 most frequent)
  - Agreement checks: (-1,0), (0,1), (-2,-1,0), (-1, 0, 1), (0, 1, 2)
  - Values (amb. class) of the attribute being disambiguated
- TiMBL parameters: k=11, MVD metric, IG weighting, IL

Evaluation

- Methodology from Śniatowski & Piasecki (2011)
  - 10 random splits into training (90%) and testing (10%) parts
  - We report values averaged across ten runs
  - Assessment of disambiguation capabilities
- IPI PAN Corpus: some tokens assigned multiple ref. tags
  - TaKIPI may output multiple tags per token
  - Weak Correctness: *%tokens* where sets of tags returned intersects with sets of tags in the gold standard
- National Corpus of Polish: always one ref. tag per token
  - Accuracy: *%tokens* tagged exactly as in gold standard

Evaluation (2)

IPI PAN Corpus

Tagger	Weak Corr.
TaKIPI	92.93%
PANTERA	92.98%
WMBT	93.16%

National Corpus of Polish 1.0

Tagger	Accuracy
PANTERA	92.95%
WMBT	92.98%

Slow tagging: 372 tokens/s (bottleneck: MBL classifier)
Fast training: 2280 tokens/s
(measured on an Intel i7 machine)

Simple majority voting

IPI PAN Corpus

Tagger	Weak Corr.
TaKIPI	92.93%
PANTERA	92.98%
WMBT	93.16%
RFTagger	89.78%
Maxent	89.70%
Best-3 3-way voting	93.96%
5-way voting	94.32%

National Corpus of Polish 1.0

Tagger	Accuracy
PANTERA	92.95%
WMBT	92.98%
RFTagger	89.09%
Voting	94.11%

Conclusions

- MBL applied to tagging Polish gives state-of-the-art results
- Practical result: WMBT, a configurable tagger for Polish
- Inclusion of WMBT boosts performance of the ensemble
- Further work:
  - More sophisticated features, long distance dependencies
  - Find optimal TiMBL parameters
  - Find optimal order of attribute disambiguation
  - Evaluate Polish taggers under 'natural' circumstances
    — joint work with Szymon Acedański (PANTERA)

Thank you for your attention

WMBT is available at http://nlp.pwr.wroc.pl/redmine/projects/wmbt/wiki

References:

Tomasz Śniatowski, Maciej Piasecki (2011). *Combining Polish Morphosyntactic Taggers*. In: Proc. of S&IIS 2011, Warsaw. Springer.