Tagery morfosyntaktyczne dla języka polskiego

Łukasz Kobyliński Witold Kieraś

Instytut Podstaw Informatyki Polskiej Akademii Nauk
ul. Jana Kazimierza 5, 01-248 Warszawa, Poland

7.12.2015

Wprowadzenie

Cele prezentacji

- podsumowanie obecnego stanu narzędzi do tagowania morfosyntaktycznego w języku polskim,
- porównanie dokładności i wykorzystywanych algorytmów z narzędziami dla innych języków europejskich,
- analiza jakościowa wyników działania poszczególnych tagerów,
- przegląd problemów, które nie zostały zaadresowane przez istniejące tagery,
- stwierdzenie, czy wśród dostępnych narzędzi dostępny jest tager o pożądanych cechach (następny slajd),
- rekomendacje dotyczące dalszych kroków.

Wprowadzenie

Cechy pożądanego tagera (M. Woliński)

Chciałbym tager, który:

- nie jest nadgorliwy, można kazać zostawić interpretacje częściowo nieujednoznacznione (np. usunąć tylko bardzo złe interpretacje),
- informuje o poziomie pewności podjętych decyzji,
- działa na niejednoznacznej segmentacji (stworzonej przez Morfeusza lub np. będącej wynikiem zastosowania po Morfeuszu słownika wyrażeń wieloczłonowych),
- daje się (względnie?) łatwo zainstalować i uruchomić na wszystkich platformach, na których jest Morfeusz,
- da się rozszerzyć o uwzględnianie informacji o czasie powstania tekstu.

Plan

1. **Tagery języka polskiego – przegląd rozwiązań**
2. **Tagery morfosyntaktyczne dla innych języków europejskich**
3. **Tagery języka polskiego – analiza ilościowa**
4. **Tagery języka polskiego – analiza jakościowa**
5. **Dyskusja i rekomendacje**

Tagery języka polskiego – przegląd rozwiązań

Czym jest tagowanie – przypomnienie

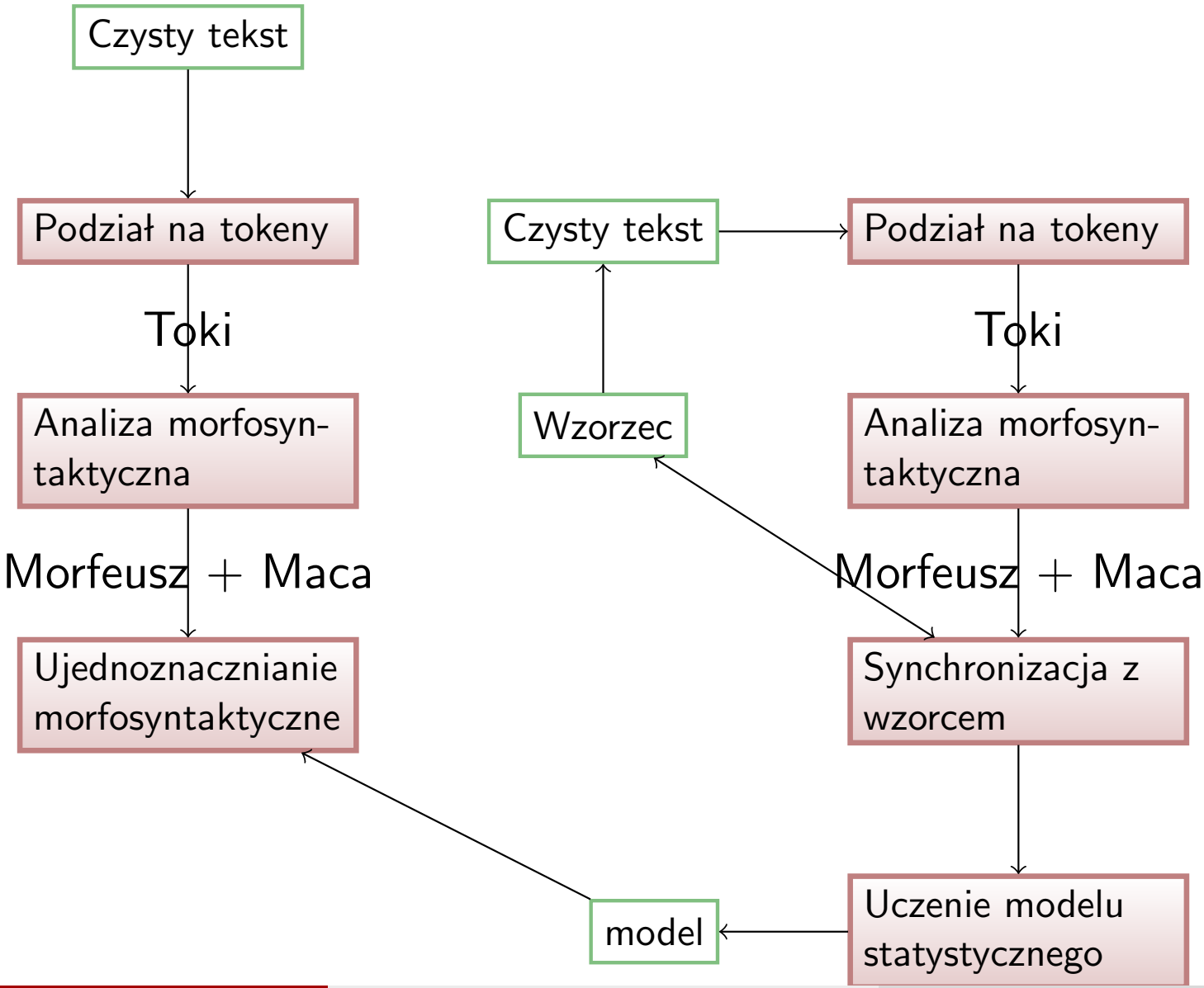
Segment (token) – wyraz lub jego fragment, znak interpunkcyjny, ciąg cyfr lub symboli. Segmenty są ciągłe oraz rozłączne.

Znacznik morfosyntaktycznzy (tag) – symbol, który można przypisać segmentowi, określający jego własności morfologiczno-składniowe.

Znakowanie morfosyntaktyczne (tagowanie) – zadanie przypisania ciągowi segmentów ciągu znaczników morfosyntaktycznych.

Segmentacja $\Rightarrow$ **Analiza morfosyntaktyczna** $\Rightarrow$ **Ujednoznacznianie morfosyntaktyczne**

Pełny stos przetwarzania



Tagery morfostynaktyczne dla języka polskiego

Tagery „archiwalne”

Dostosowane do tagsetu IPI PAN, modele uczone na korpusie IPI.

- tager Ł. Dębowskiego – statystyczny tager trigramowy,
  weak correctness $= 90,59\%$
- TaKIPI – tager hybrydowy, oparty na drzewach decyzyjnych, częstości unigramów i ręcznie utworzonych regułach.
  weak correctness $= 91.30\%$

Tagery morfostynaktyczne dla języka polskiego

Tagery uwzględniające tagset NKJP

- Pantera [Acedański 2010] – adaptacja algorytmu Brilla do języków bogatych morfologicznie, takich jak polski,
- WMBT [Radziszewski and Śniatowski 2011] – tager oparty na uczeniu pamięciowym, rozbudowany o wielowarstwowość dla uwzględnienia wielu atrybutów znakowania w języku polskim,
- Concraft [Waszczuk 2012] – tager warstwowy, oparty na Conditional Random Fields (CRF); wyniki dezambiguacji morfosyntaktycznej przekazywane są z jednej warstwy do drugiej,
- WCRFT [Radziszewski 2013] – również oparty na CRF; osobne modele wykorzystywane są do dezambiguacji poszczególnych atrybutów opisu morfosyntaktycznego,

Tagery morfostynaktyczne dla języka polskiego

Modele dla tagerów zaimplementowanych dla innych języków

- TnT Tagger – statystyczny tager trigramowy, model dla PL przygotowany przez M. Miłkowskiego, dokładność „ok. 88%”,
  `http://zil.ipipan.waw.pl/NKJP%20model%20for%20TnT%20Tagger`
- OpenNLP – tager maksimum entropii, model dla PL przygotowany przez P. Pęzika (nie jest udostępniony publicznie).
  `http://clarin.pelcra.pl/tools/tagger`

Tager Brilla – zasada działania

Uczenie tagera

- wytrenuj tager unigramowy na podstawie zbioru uczącego,
- otaguj zbiór rozwojowy za pomocą tagera unigramowego,
- iteracyjnie znajdź transformacje, które mogą poprawić największą liczbę błędów, wprowadzając jednocześnie jak najmniej pomyłek
- zachowaj zbiór najlepszych transformacji – model.

r	good(r)	bad(r)
Zmień przypadek przyimka z *acc* na *loc*, if jeśli kończy się na *na* i jeden z kolejnych tokenów jest w przypadku *loc*.	2496	113
Zmień przypadek przymiotnika z *loc* na *inst*, jeśli jeden z kolejnych tokenów ma przypadek *inst* i kończy się na *em*.	921	29

Tager Brilla – przykładowe reguły

Transformacje Zachodzą według jednego z szablonów:

- $t_i := A$ *if* $t_i = B \wedge \exists_{o \in O_1} t_{i+o} = C$
- $t_i := A$ *if* $t_i = B \wedge \forall_{o \in O_2} t_{i+o} = D$
- $t_i := A$ *if* $t_i = B$ i $i-$te słowo jest z wielkiej litery
- $t_i := A$ *if* $t_i = B$ i $(i-1)-$te słowo jest z wielkiej litery

gdzie:

- $O_1 \in \{\{1\}, \{-1\}, \{2\}, \{-2\}, \{1,2\}, \{-1,-2\}, \{1,2,3\}, \{-1,-2,-3\}\}$,
- $O_2 \in \{\{-2,-1\}, \{-1,1\}, \{1,2\}\}$,
- A, B, C, D – tagi.

Tager CRF – zasada działania

TODO

Przenośność i łatwość wykorzystania

- **Concraft** instalowany i uruchamiany z wykorzystaniem Haskell Platform, która dostępna jest pod wszystkie główne systemy operacyjne,
- **WCRFT, Pantera** – wymagają kompilacji, proces kompilacji dostosowany do środowiska Linuksowego,
- **WMBT** – Python.

Concraft, WCRFT, WMBT – silnie zależą od stosu Corpus2 / Toki / Maca, których kompilacja pod Windows jest możliwa, ale nietrywialna (Visual Studio).

Tagery morfosyntaktyczne dla innych języków europejskich

Tagowanie języka angielskiego

System	Metoda	Publikacja	Dokładność
BI-LSTM-CRF	Bidirectional LSTM-CRF Model	Huang et al. (2015)	97.55
SCCN	Semi-supervised condensed nearest neighbor	Søgaard (2011)	97.50
Morče/COMPOST	Averaged Perceptron	Spoustová et al. (2009)	97.44
structReg	CRFs with structure regularization	Sun(2014)	97.36
LTAG-spinal	Bidirectional perceptron learning	Shen et al. (2007)	97.33
Stanford Tagger 2.0	Maximum entropy cyclic dependency network	Manning (2011)	97.32
Stanford Tagger 2.0	Maximum entropy cyclic dependency network	Manning (2011)	97.29
Stanford Tagger 1.0	Maximum entropy cyclic dependency network	Toutanova et al. (2003)	97.24

Tagowanie języka angielskiego

System	Metoda	Publikacja	Dokładność
Morče/COMPOST	Averaged Perceptron	Spoustová et al. (2009)	97.23
LAPOS	Perceptron based training with lookahead	Tsuruoka, Miyao, and Kazama (2011)	97.22
SVMTool	SVM-based tagger and tagger generator	Giménez and Márquez (2004)	97.16
Maxent easiest-first	Maximum entropy bidirectional easiest-first inference	Tsuruoka and Tsujii (2005)	97.15
Averaged Perceptron	Averaged Perception discriminative sequence model	Collins (2002)	97.11
GENiA Tagger**	Maximum entropy cyclic dependency network	Tsuruoka, et al (2005)	97.05
MElt	MEMM with external lexical information	Denis and Sagot (2009)	96.96
TnT*	Hidden markov model	Brants (2000)	96.46

Tagery języków europejskich – porównanie

Tager	Język	Rozmiar tagsetu	Korpus treningowy	Dokładność
Concraft	polski	4 000 / 1 000	1M	91,07%
Obeliks	słoweński	1 903	500k	91,34%
Morče	czeski	3 922 / 1 571	2M	95,67%
Featurama	czeski	3 922 / 1 571	2M	95,66%
Morphodita	czeski	3 922 / 1 571	2M	95,75%
BI-LSTM-CRF	angielski	36+12	1M	97,55%

Tagery języka polskiego – analiza ilościowa

Metoda ewaluacji

Miara jakości znakowania

- ze względu na możliwość wystąpienia różnic w segmentacji pomiędzy wynikiem znakowania, a złotym standardem, wykorzystujemy dolne ograniczenie trafności (accuracy lower bound, Acc_{lower}) do oceny dokładności tagerów,
- miara ta karze wszelkie zmiany segmentacyjne w stosunku do złotego standardu i traktuje takie tokeny jako sklasyfikowane błędnie,
- token traktowany jest jako oznakowany prawidłowo, jeśli zbiór jego interpretacji ma niepuste przecięcie ze zbiorem interpretacji zwracanych przez tager,
- niezależne sprawdzamy dokładność dla znanych (Acc^{K}_{lower}) i nieznanych słów (Acc^{U}_{lower}), aby ocenić skuteczność ew. modułów odgadywania.

Ewaluacja pojedynczych tagerów

Eksperymenty na milionowym podkorpusie Narodowego Korpusu Języka Polskiego, ver. 1.1, 10-krotna walidacja krzyżowa.

n	Tager	Acc_{lower}	Acc^{K}_{lower}	Acc^{U}_{lower}
1	Pantera	88.95%	91.22%	15.19%
2	WMBT	90.33%	91.26%	60.25%
3	WCRFT	90.76%	91.92%	53.18%
4	Concraft	91.07%	92.06%	58.81%

- Acc_{lower} – łączna dokładność,
- Acc^{K}_{lower} – dokładność dla znanych słów,
- Acc^{U}_{lower} – dokładność dla słów nieznanych.

Analiza rezultatu działania tagerów

Porównanie wyników

- Wszystkie zwracają prawidłowy tag: **82,78%**
  unikam fin:sg:pri:imperf

  fin:sg:pri:imperf+ fin:sg:pri:imperf+ fin:sg:pri:imperf+ fin:sg:pri:imperf+
- Większość zwraca prawidłowy tag: **7,95%**
  kapitalistów subst:pl:gen:m1

  subst:pl:gen:m1+ subst:pl:gen:m1+ subst:pl:gen:m1+ subst:pl:acc:m1-
- Równowaga w głosowaniu: **2,71%**
  powolny adj:sg:nom:m3:pos

  adj:sg:nom:m3:pos+ adj:sg:nom:m3:pos+ adj:sg:acc:m3:pos- adj:sg:acc:m3:pos-
- Prawidłowy tag w mniejszości: **2,38%**

  twarzy subst:sg:loc:f subst:sg:gen:f- subst:sg:gen:f- subst:sg:gen:f- subst:sg:loc:f+
- Wszystkie się mylą: **4.18%**
  biurka subst:pl:nom:n subst:pl:acc:n- subst:pl:acc:n- subst:sg:gen:n- subst:pl:acc:n-
  (Peggy) McCreary subst:sg:nom:f

  subst:sg:gen:f- subst:sg:gen:n- subst:sg:nom:n- subst:sg:acc:m1-

Podział na klasy gramatyczne

		Acc_{lower} (%)			
klasa	liczność	PANTERA	WMBT	WCRFT	Concraft
subst	331570	85,21	86,25	87,36	88,29
interp	223542	99,63	99,97	99,97	99,97
adj	128703	76,53	81,10	81,56	82,52
prep	115818	97,04	97,28	97,54	98,05
qub	68079	92,98	93,82	92,91	92,92
fin	59458	98,64	98,70	98,81	98,94
praet	53326	90,90	88,96	89,80	89,69
conj	44840	95,17	95,41	94,61	93,96
adv	42750	95,31	95,59	95,29	94,77
inf	19213	98,91	99,20	99,09	99,14
comp	17842	97,26	97,29	96,84	96,88
num	16160	33,40	56,40	60,32	55,99

Najczęstsze błędy

Concraft: najczęstsze błędy wyboru klasy gramatycznej

Concraft	NKJP	liczność	wystąpienia (%)
adj	subst	199	0.1654
conj	qub	178	0.1479
subst	adj	162	0.1346
adv	qub	159	0.1321
subst	ger	152	0.1263
qub	conj	151	0.1255
subst	brev	140	0.1164
ger	subst	128	0.1064
num	adj	108	0.0898
ppas	adj	108	0.0898
qub	adv	91	0.0756
adj	ppas	71	0.0590
adj	num	71	0.0590

Najczęstsze błędy

Concraft: najczęstsze błędy doboru tagu w ramach klasy *subst*

Concraft	NKJP	liczność	wystąpienia (%)
sg:nom:m3	sg:acc:m3	191	0.1587
sg:acc:m3	sg:nom:m3	153	0.1272
sg:acc:n	sg:nom:n	134	0.1114
sg:nom:n	sg:acc:n	117	0.0972
pl:nom:m3	pl:acc:m3	89	0.0740
pl:acc:f	pl:nom:f	78	0.0648
pl:nom:f	pl:acc:f	71	0.0590
pl:acc:m3	pl:nom:m3	68	0.0565
sg:gen:m1	sg:acc:m1	67	0.0557
sg:acc:m1	sg:gen:m1	53	0.0440
pl:nom:n	pl:acc:n	48	0.0399
sg:gen:f	pl:gen:f	47	0.0391

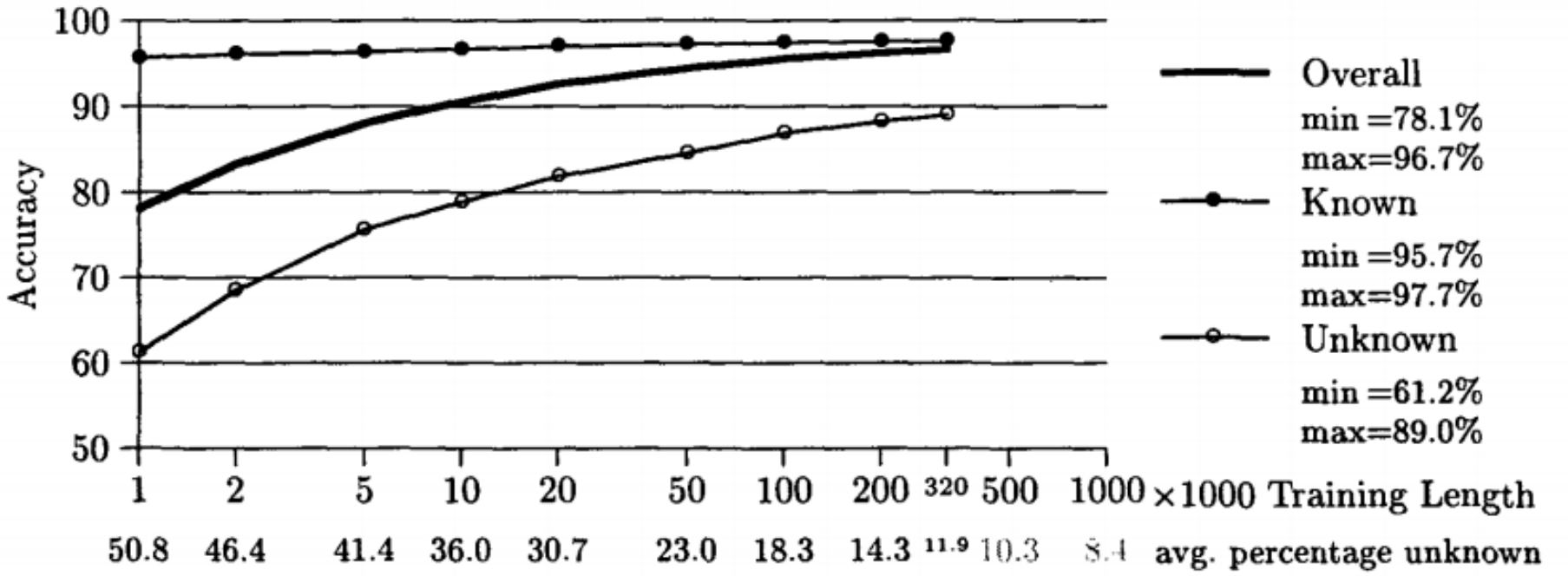
Najczęstsze błędy

Concraft: najczęstsze błędy doboru tagu w ramach klasy *adj*

Concraft	NKJP	liczność	wystąpienia (%)
sg:nom:m3:pos	sg:acc:m3:pos	90	0.0748
sg:acc:m3:pos	sg:nom:m3:pos	72	0.0598
sg:nom:m1:pos	sg:nom:m3:pos	57	0.0474
sg:nom:n:pos	sg:acc:n:pos	51	0.0424
pl:nom:m3:pos	pl:acc:m3:pos	49	0.0407
pl:nom:f:pos	pl:acc:f:pos	46	0.0382
pl:acc:f:pos	pl:nom:f:pos	44	0.0366
sg:nom:m3:pos	sg:nom:m1:pos	42	0.0349
sg:acc:n:pos	sg:nom:n:pos	32	0.0266
pl:acc:m3:pos	pl:nom:m3:pos	28	0.0233
pl:nom:n:pos	pl:acc:n:pos	27	0.0224
pl:nom:m3:pos	pl:nom:f:pos	25	0.0208
pl:acc:n:pos	pl:nom:n:pos	21	0.0175

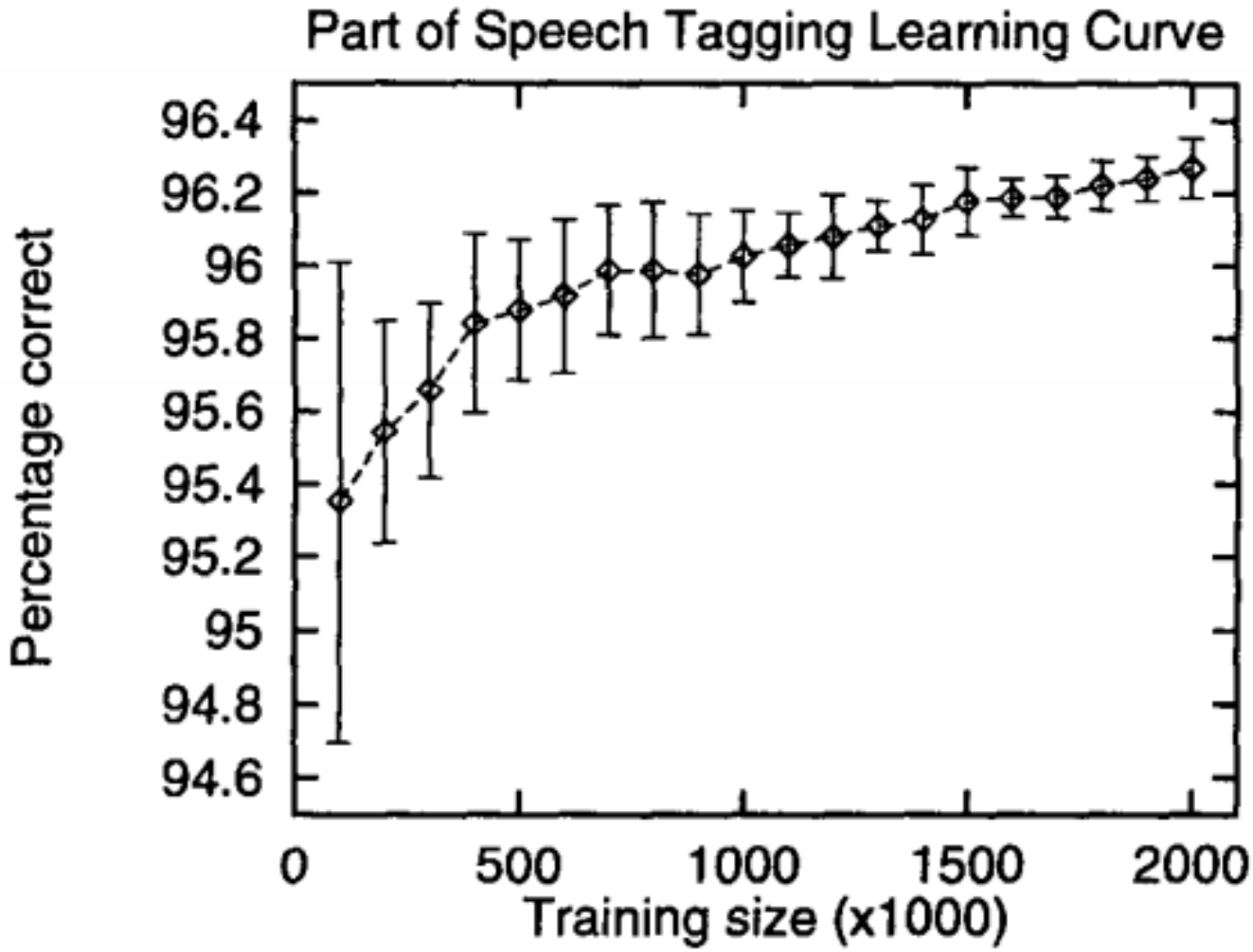
Rozmiar danych treningowych

TnT Tagger – NEGRA corpus, 30 000 tokenów testowych.



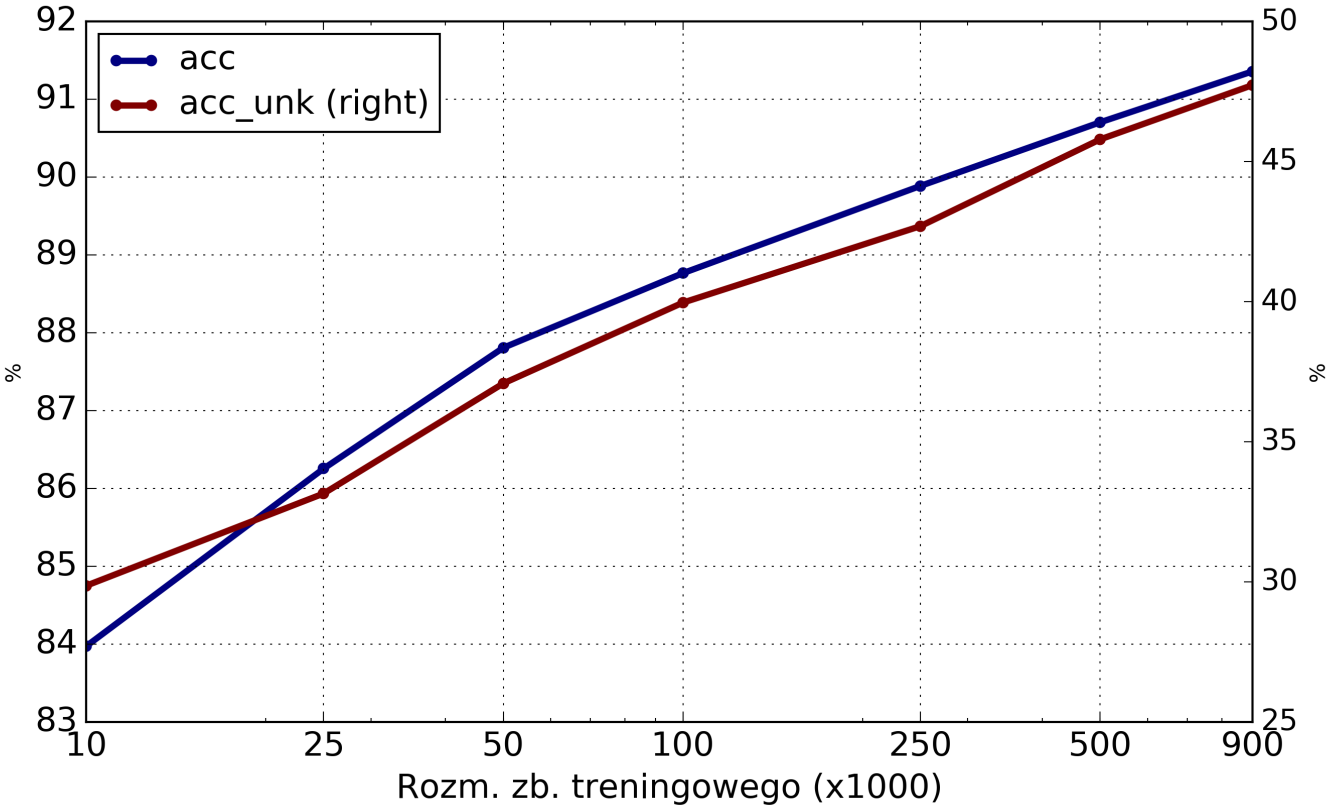
Rozmiar danych treningowych

MBT Tagger – WSJ corpus, kroswalidacja krzyżowa



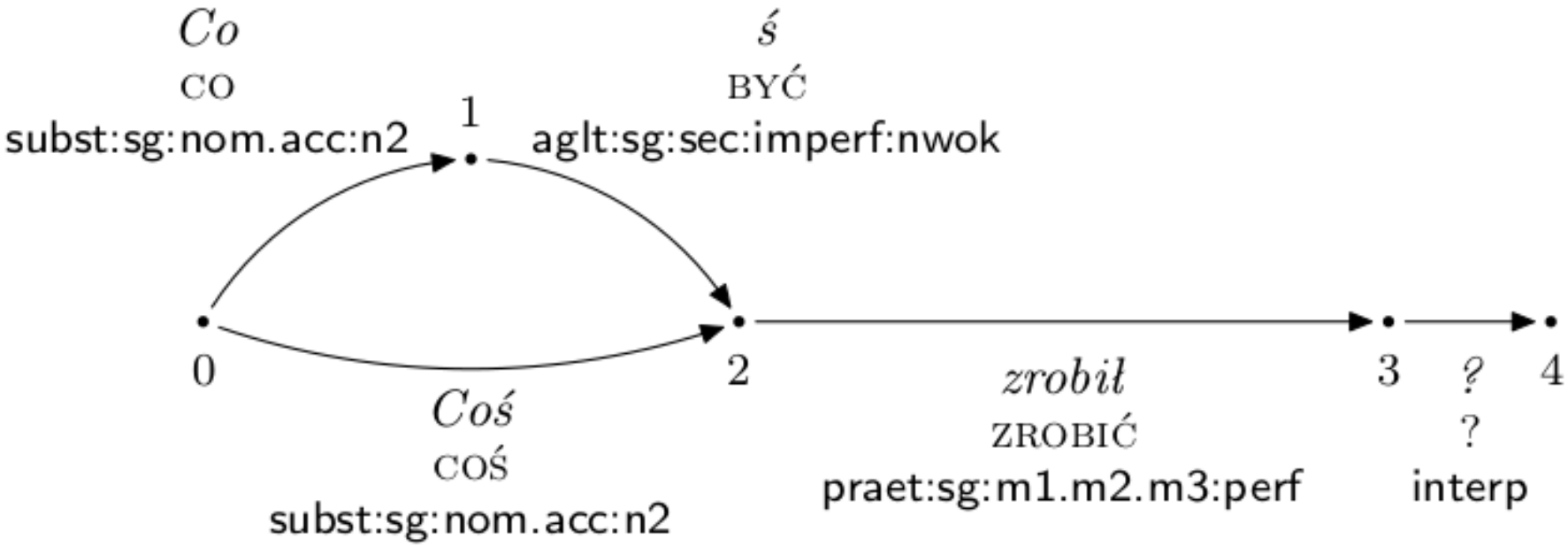
Rozmiar danych treningowych

Concraft – NKJP 1M, 100 000 tokenów testowych.



Problem niejednoznaczności segmentacji

Na czym polega problem?



Problem niejednoznaczności segmentacji

Jak często występują niejednoznaczności?

Korpus NKJP 1M, Morfeusz 1 SGJP:

645 wystąpień na 1 095 118 segmentów (0.0589%)

kiedyś	234 / 1
gdzieś	172 / 1
miałem	99 / 98
udziałem	40 / 0
musiałem	28 / 28
sms-a	6 / 0
działam	6 / 0
doń	5 / 4
tyłem	4 / 0
pis-em	4 / 0
podziałem	3 / 0
piekłem	3 / 0
czekałem	3 / 3
jadłem	3 / 3

pis-owi	3
winnym	2
prl-em	2
wyłom	2
rozdziałem	2
hiv-em	2
pit-ów	2
działem	1
rop-em	1
tir-a	1
kor-owcy	1
urm-em	1
kor-em	1
dj-a	1

ipn-em	1
sms-ów	1
pgr-ach	1
zus-em	1
vat-em	1
siadłem	1
msz-ów	1
zoz-ów	1
mosir-em	1
vip-om	1
msz-ecie	1
zoz-owi	1
czemuś	1

Problem niejednoznaczności segmentacji

Jak często występują niejednoznaczności?

Próbka 100M NKJP, Morfeusz 1 SGJP:

40 354 wystąpień na 101 052 527 segmentów (0.0399%)

kiedyś	12751	czemuś	171	pit-ów	65
miałem	8350	działam	153	Łks-em	60
gdzieś	6171	działem	151	pis-em	44
udziałem	4988	piekłem	130	siadłem	43
musiałem	2173	zus-em	95	skok-i	39
czekałem	537	sms-em	95	gks-em	39
tyłem	523	tir-ów	93	padłem	36
doń	414	zoz-ów	86	pgr-ów	30
podziałem	411	tir-a	85	vip-a	30
sms-a	357	zus-owi	82	pis-owi	28
vip-ów	305	azs-em	81	skok-ów	24
winnym	256	vat-em	76	pks-em	24
sms-ów	207	rozdziałem	76	dj-ów	20
jadłem	199	wyłom	65	dj-e	20

Problem niejednoznaczności segmentacji

Jak często występują niejednoznaczności?

Korpus NKJP 1M, Morfeusz 2:

2583 wystąpień na 1 095 118 segmentów (0.2359%)

coś	777
ktoś	382
czym	334
kiedyś	234
gdzieś	172
miałem	99
czegoś	97
kogoś	82
czymś	63
kimś	42
gdybym	40
udziałem	40

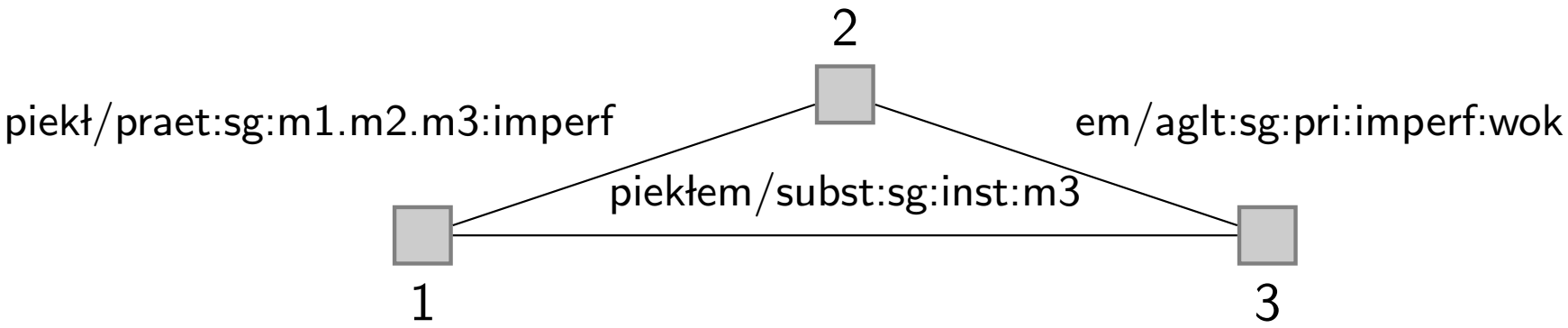
komuś	31
gdybyśmy	19
tom	19
gdybyś	7
działam	6
jam	5
doń	5
oścież	5
gdybyście	4
tyłem	4
musiałem	3
podziałem	3
piekłem	3

czekałem	3
jadłem	3
rozdziałem	2
wyłom	2
bom	2
coście	1
czyżbyś	1
czyżem	1
siadłem	1
czemuś	1
działem	1

Problem niejednoznaczności segmentacji

Możliwe rozwiązania: tagset pośredni (A. Radziszewski)

Wprowadźmy tagset pośredni, który pozwoli uniknąć części niejednoznaczności



piekłem

piec fin:sg:m1:pri:imperf:prt
piec fin:sg:m2:pri:imperf:prt
piec fin:sg:m3:pri:imperf:prt
piekło subst:sg:inst:n

Problem niejednoznaczności segmentacji

Możliwe rozwiązania: tagset pośredni

Rozwiązanie analogiczne do trypu *−p composite* w Morfeuszu 2:

```
$ echo piekłem | morfeusz_analyzer
[0,1,piekł,piec:v,praet:sg:m1.m2.m3:imperf,_,_]
[0,2,piekłem,piekło,subst:sg:inst:n2,pospolita,_]
[1,2,em,być,aglt:sg:pri:imperf:wok,_,_]

$ echo piekłem | morfeusz_analyzer −p composite
[0,1,piekłem,piec:v,praet:sg:m1.m2.m3:pri:imperf,_,_
0,1,piekłem,piekło,subst:sg:inst:n2,pospolita,_]
```

Problem niejednoznaczności segmentacji

Możliwe rozwiązania: dostosowanie tagera do przetwarzania DAGów

Na przykład, dla tagera Concraft: konieczna jest modyfikacja całego stosu przetwarzania:

- modyfikacja istniejących formatów zapisu korpusów (XML, Plain),
- modyfikacja narzędzi: Corpus2, Maca,
- reimplementacja algorytmu w tagerze, aby był w stanie przetwarzać dane w reprezentacji grafowej.

Poziom pewności ujednoznaczniania morfosyntaktycznego

Oczekujemy, że: tager informuje o poziomie pewności podjętych decyzji

Tagery oparte na metodach uczenia maszynowego mogą zwracać prawdopodobieństwa brzegowe wyboru poszczególnych interpretacji.

Implementacja w Concraft:

$$\$\tilde{}/.cabal/bin/concraft - pl\ tag\ - m\ model.gz\ <\ test.plain$$

rzucała space
    rzucać praet:sg:f:imperf 1.000
światło space
    światło adv:pos 0.000
    światło subst:sg:acc:n 0.929
    światło subst:sg:nom:n 0.071
    światło subst:sg:voc:n 0.000

tylko space
    Tylka subst:sg:voc:f 0.000
    tylko conj 0.209
    tylko qub 0.791
na space
    na interj 0.000
    na prep:acc 1.000
    na prep:loc 0.000
podłogę space
    podłoga subst:sg:acc:f 1.000

Problem lematyzacji

Tagery języka polskiego – analiza jakościowa

TODO

Dyskusja i rekomendacje

Różne tagsety dla języka polskiego?

Obecnie funkcjonują równolegle dwa tagsety języka polskiego

- tagset NKJP, używany do anotacji korpusu, a także w większości innych zasobów językowych,
- tagset Morfeusza.

Skutkuje to sytuacją, w której w sposób niejawny dokonywana jest ciągła konwersja pomiędzy tagsetami:

```
tagset_from=sgjp
tagset_to=nkjp
override=n1:n
override=n2:n
override=n3:n
override=p1:m1
override=p2:n
override=p3:n
```

```
tagset_from=morfeusz2; tagset_to=nkjp
override=dig:num; override=nie:conj
override=romandig:num
override=prefa:ign
override=prefppas:ign
override=prefs:ign; override=prefv:ign
override=naj:ign; override=cond:ign
override=substa:ign
```

Struktura danych treningowych w NKJP1M

TODO

Podsumowanie

Cechy pożądanego tagera (M. Woliński)

TODO Chciałbym tager, który:

- nie jest nadgorliwy, można kazać zostawić interpretacje częściowo nieujednoznacznione (np. usunąć tylko bardzo złe interpretacje), obecnie: każdy
- informuje o poziomie pewności podjętych decyzji, obecnie: Concraft, możliwe: WCRFT, WMBT
- działa na niejednoznacznej segmentacji (stworzonej przez Morfeusza lub np. będącej wynikiem zastosowania po Morfeuszu słownika wyrażeń wieloczłonowych), obecnie: tylko tagset pośredni, możliwe: w każdym po modyfikacjach całego stosu

Podsumowanie (2)

Cechy pożądanego tagera (M. Woliński)

TODO Chciałbym tager, który:

- daje się (względnie?) łatwo zainstalować i uruchomić na wszystkich platformach, na których jest Morfeusz, obecnie: ograniczeniem jest Maca
- da się rozszerzyć o uwzględnianie informacji o czasie powstania tekstu. obecnie: żaden, możliwe: w każdym jako cecha ML

Podziękowania

Podziękowania za sugestie i uwagi dla:

- Adama Radziszewskiego
- Jakuba Waszczuka
- Szymona Acedańskiego

Dziękujemy za uwagę!

Bibliografia I

Acedański, Szymon, 2010.
A morphosyntactic Brill tagger for inflectional languages.
In Advances in Natural Language Processing.

Brill, Eric and Jun Wu, 1998.
Classifier combination for improved lexical disambiguation.
In Proceedings of the 17th international conference on Computational linguistics - Volume 1, COLING '98. Stroudsburg, PA, USA: Association for Computational Linguistics.

Bibliografia II


Śniatowski, Tomasz and Maciej Piasecki, 2012.
Combining Polish morphosyntactic taggers.
In Pascal Bouvry, Mieczysław A. Kłopotek, Franck Leprévost, Małgorzata Marciniak, Agnieszka Mykowiecka, and Henryk Rybiński (eds.), Security and Intelligent Information Systems, volume 7053 of LNCS. Springer-Verlag.


Radziszewski, Adam, 2013.
A tiered CRF tagger for Polish.
In R. Bembenik, Ł. Skonieczny, H. Rybiński, M. Kryszkiewicz, and M. Niezgódka (eds.), Intelligent Tools for Building a Scientific Information Platform: Advanced Architectures and Solutions. Springer Verlag.

Bibliografia III


Radziszewski, Adam and Szymon Acedański, 2012.
Taggers gonna tag: an argument against evaluating disambiguation capacities of morphosyntactic taggers.
In Proceedings of TSD 2012, LNCS. Springer-Verlag.


Radziszewski, Adam and Tomasz Śniatowski, 2011a.
A Memory-Based Tagger for Polish.
In Proceedings of the LTC 2011.


van Halteren, Hans, Walter Daelemans, and Jakub Zavrel, 2001.
Improving accuracy in word class tagging through the combination of machine learning systems.
Comput. Linguist., 27(2):199–229.

Bibliografia IV

Waszczuk, Jakub, 2012.
Harnessing the CRF complexity with domain-specific constraints. The case of morphosyntactic tagging of a highly inflected language.
In Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012). Mumbai, India.