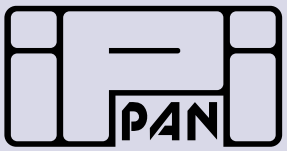
Konwerter tagsetów TaCo

Bartosz Zaborowski



INSTITUTE OF COMPUTER SCIENCE
POLISH ACADEMY OF SCIENCES
ul. Jana Kazimierza 5, 01-248 Warszawa

11 lutego 2013

Motywacja

- Do anotacji korpusów używane są różne typy tagów.
- Tagsety z reguły różnią się pomiędzy korpusami (również pomiędzy korpusami dla tego samego języka).
- Narzędzia i zasoby NLP często są przystosowane do konkretnego tagsetu.
- (Re-)anotacja ręczna: kosztowna.
- Automatyczne (re-)anotowanie zwykłym taggerem: kiepska jakość (niezależnie od jakości pierwotnej anotacji).
- Ręcznie pisane reguły: obie powyższe wady (w różnych stopniach).

Motywacja

- Do anotacji korpusów używane są różne typy tagów.
- Tagsety z reguły różnią się pomiędzy korpusami (również pomiędzy korpusami dla tego samego języka).
- Narzędzia i zasoby NLP często są przystosowane do konkretnego tagsetu.
- (Re-)anotacja ręczna: kosztowna.
- Automatyczne (re-)anotowanie zwykłym taggerem: kiepska jakość (niezależnie od jakości pierwotnej anotacji).
- Ręcznie pisane reguły: obie powyższe wady (w różnych stopniach).

Konwersja regułowa

- DZ Interset (Daniel Zeman):
  - proste reguły (kilka-kilkadziesiąt godzin pracy na tagset),
  - Interset – tagset pośredni (jak Interlingua w tłumaczeniu),
  - ignoruje kontekst, wyjątki gramatyczne i inne wskazówki.

- nieoficjalny(?) konwerter KIPI->NKJP dla Korpusu Słownika Frekwencyjnego (Michał Lenart i inni):
  - dla konkretnej pary tagsetów (i trochę korpusu),
  - długie listy słów i fraz traktowanych w szczególny sposób.

Konwersja regułowa

- DZ Interset (Daniel Zeman):
  - proste reguły (kilka-kilkadziesiąt godzin pracy na tagset),
  - Interset – tagset pośredni (jak Interlingua w tłumaczeniu),
  - ignoruje kontekst, wyjątki gramatyczne i inne wskazówki.

- nieoficjalny(?) konwerter KIPI->NKJP dla Korpusu Słownika Frekwencyjnego (Michał Lenart i inni):
  - dla konkretnej pary tagsetów (i trochę korpusu),
  - długie listy słów i fraz traktowanych w szczególny sposób.

Opis zadania

- Dla danego korpusu otagowanego równolegle za pomocą dwóch różnych tagsetów (źródłowego i docelowego)...
- Zadanie polega na zbudowaniu automatycznej, statystycznej metody konwersji...
- która dla danego słowa wraz z kontekstem otagowanym tagsetem źródłowym znajdzie najlepiej pasujący tag z tagsetu docelowego.

Opis zadania

- Dla danego korpusu otagowanego równolegle za pomocą dwóch różnych tagsetów (źródłowego i docelowego)...
- Zadanie polega na zbudowaniu automatycznej, statystycznej metody konwersji...
- która dla danego słowa wraz z kontekstem otagowanym tagsetem źródłowym znajdzie najlepiej pasujący tag z tagsetu docelowego.

Opis zadania

- Dla danego korpusu otagowanego równolegle za pomocą dwóch różnych tagsetów (źródłowego i docelowego)...
- Zadanie polega na zbudowaniu automatycznej, statystycznej metody konwersji...
- która dla danego słowa wraz z kontekstem otagowanym tagsetem źródłowym znajdzie najlepiej pasujący tag z tagsetu docelowego.

Tagsety użyte przy pracy nad konwerterem

- źródłowy: tagset korpusu IPI
- docelowy: tagset NKJP

Przykład:

forma ort.	tag źródłowy	tag docelowy
To	pred	pred
dużo	adv:pos	num:pl:nom:f:rec
czy	conj	qub
mało	qub	num:pl:nom:f:rec
?	interp	interp

Baseline

- Użyjemy najprostszego taggera – unigramowego.
- Trzeba go dostosować do zadania: częstości będą liczone dla tagów źródłowych zamiast dla słów.

Wybór klasyfikatora

- Algorytm baseline jest najprostszym klasyfikatorem. Czemu nie spróbować jakiegoś lepszego?
- Przetestowałem 45 klasyfikatorów z biblioteki WEKA.
- Dane wejściowe: stabelaryzowany korpus (jeden rekord = kodowane pozycyjnie tagi źródłowe dla słowa i kontekstu (+/- 1 słowo); atrybut klasy = tag docelowy).
- Zwycięzca testu: J48 (= javowa implementacja C4.5).
- Ostatecznie wybrałem C5.0 (następce C4.5) ze względu na wydajność.

Example (zwykła i pozycyjna forma tagu)

subst:sg:acc:n -,-,-,-,acc,-,n,-,sg,-,subst,-,-

Pojedynczy rekord tabeli na obecnym etapie:

pozycyjny tag –1	tag pozycyjny	tag pozycyjny +1	tag

Wybór klasyfikatora

- Algorytm baseline jest najprostszym klasyfikatorem. Czemu nie spróbować jakiegoś lepszego?
- Przetestowałem 45 klasyfikatorów z biblioteki WEKA.
- Dane wejściowe: stabelaryzowany korpus (jeden rekord = kodowane pozycyjnie tagi źródłowe dla słowa i kontekstu (+/- 1 słowo); atrybut klasy = tag docelowy).
- Zwycięzca testu: J48 (= javowa implementacja C4.5).
- Ostatecznie wybrałem C5.0 (następce C4.5) ze względu na wydajność.

Example (zwykła i pozycyjna forma tagu)

subst:sg:acc:n -,-,-,-,acc,-,n,-,sg,-,subst,-,-

Pojedynczy rekord tabeli na obecnym etapie:

pozycyjny tag –1	tag pozycyjny	tag pozycyjny +1	tag

Wybór klasyfikatora

- Algorytm baseline jest najprostszym klasyfikatorem. Czemu nie spróbować jakiegoś lepszego?
- Przetestowałem 45 klasyfikatorów z biblioteki WEKA.
- Dane wejściowe: stabelaryzowany korpus (jeden rekord = kodowane pozycyjnie tagi źródłowe dla słowa i kontekstu (+/- 1 słowo); atrybut klasy = tag docelowy).
- Zwycięzca testu: J48 (= javowa implementacja C4.5).
- Ostatecznie wybrałem C5.0 (następce C4.5) ze względu na wydajność.

Example (zwykła i pozycyjna forma tagu)

subst:sg:acc:n	-,-,-,-,acc,-,n,-,sg,-,subst,-,-

Pojedynczy rekord tabeli na obecnym etapie:

pozycyjny tag –1	tag pozycyjny	tag pozycyjny +1	tag

Kontekst i pozycyjne kodowanie tagów

- Rozmiar kontekstu +/- 1 słowo użyty przed chwilą był wybrany arbitralnie.
- Po przetestowaniu różnych konfiguracji. . .
- Lewy kontekst z 3 słowami i prawy z 1 słowem okazał się najlepszy.

- Założenie: oba tagsety są pozycyjne i (potencjalnie) duże.
- Nie wszystkie tagi występują w korpusie, większość 3-gramów tagów nie występuje. Zatem tagi zostały podzielone.
- Testy zaskoczyły: zwykła tekstowa forma tagów źródłowych dla badanego słowa działa lepiej.
- Mniej zaskakujące: tagów docelowych też lepiej nie dzielić.

Pojedynczy rekord tabeli na obecnym etapie:

p. tag –3	p. tag –2	p. tag –1	p. tag	p. tag +1	tag

Kontekst i pozycyjne kodowanie tagów

- Rozmiar kontekstu +/- 1 słowo użyty przed chwilą był wybrany arbitralnie.
- Po przetestowaniu różnych konfiguracji...
- Lewy kontekst z 3 słowami i prawy z 1 słowem okazał się najlepszy.

- Założenie: oba tagsety są pozycyjne i (potencjalnie) duże.
- Nie wszystkie tagi występują w korpusie, większość 3-gramów tagów nie występuje. Zatem tagi zostały podzielone.
- Testy zaskoczyły: zwykła tekstowa forma tagów źródłowych dla badanego słowa działa lepiej.
- Mniej zaskakujące: tagów docelowych też lepiej nie dzielić.

Pojedynczy rekord tabeli na obecnym etapie:

p. tag –3	p. tag –2	p. tag –1	p. tag	p. tag +1	tag

Kontekst i pozycyjne kodowanie tagów

- Rozmiar kontekstu +/- 1 słowo użyty przed chwilą był wybrany arbitralnie.
- Po przetestowaniu różnych konfiguracji...
- Lewy kontekst z 3 słowami i prawy z 1 słowem okazał się najlepszy.

- Założenie: oba tagsety są pozycyjne i (potencjalnie) duże.
- Nie wszystkie tagi występują w korpusie, większość 3-gramów tagów nie występuje. Zatem tagi zostały podzielone.
- Testy zaskoczyły: zwykła tekstowa forma tagów źródłowych dla badanego słowa działa lepiej.
- Mniej zaskakujące: tagów docelowych też lepiej nie dzielić.

Pojedynczy rekord tabeli na obecnym etapie:

p. tag –3	p. tag –2	p. tag –1	p. tag	p. tag +1	tag

Kontekst i pozycyjne kodowanie tagów

- Rozmiar kontekstu +/- 1 słowo użyty przed chwilą był wybrany arbitralnie.
- Po przetestowaniu różnych konfiguracji...
- Lewy kontekst z 3 słowami i prawy z 1 słowem okazał się najlepszy.

- Założenie: oba tagsety są pozycyjne i (potencjalnie) duże.
- Nie wszystkie tagi występują w korpusie, większość 3-gramów tagów nie występuje. Zatem tagi zostały podzielone.
- Testy zaskoczyły: zwykła tekstowa forma tagów źródłowych dla badanego słowa działa lepiej.
- Mniej zaskakujące: tagów docelowych też lepiej nie dzielić.

Pojedynczy rekord tabeli na obecnym etapie:

p. tag –3	p. tag –2	p. tag –1	tag	p. tag +1	tag

Informacje zawarte w formie ortograficznej

- Bezpośrednie użycie formy ortograficznej jest niewykonalne.
- Najważniejsze części słowa dla języka polskiego: prefiksy, sufiksy i wielkość pierwszej litery (WPL).
- Ostatecznie:
  WPL dla każdego słowa (kontekst + przetwarzane słowo),
  1-literowy prefiks przetwarzanego słowa,
  1-, 2- i 3-literowe sufiksy przetwarzanego słowa.

Pojedynczy rekord tabeli na obecnym etapie:

WPL(–3..+1)	prefiks(1)	sufiks(1,2,3)	PT(–3..–1)	T	PT(+1)	T

Informacje zawarte w formie ortograficznej

- Bezpośrednie użycie formy ortograficznej jest niewykonalne.
- Najważniejsze części słowa dla języka polskiego: prefiksy, sufiksy i wielkość pierwszej litery (WPL).
- Ostatecznie:
  WPL dla każdego słowa (kontekst + przetwarzane słowo),
  1-literowy prefiks przetwarzanego słowa,
  1-, 2- i 3-literowe sufiksy przetwarzanego słowa.

Pojedynczy rekord tabeli na obecnym etapie:

WPL(–3..+1)	prefiks(1)	sufiks(1,2,3)	PT(–3..–1)	T	PT(+1)	T

Dodatkowe atrybuty wyuczone na podstawie błędów

- Nie możemy wykorzystać formy ortograficznej dla wszystkich słów, ale dla małej liczby wyjątków gramatycznych możemy.
- Jak je znaleźć?
- Klasyfikator odnajduje reguły/regularności. Myli się na wyjątkach.
- Rozwiązanie:

wyjątek?(−3..+1)	WPL(−3..+1)	P(1)	S(1,2,3)	PT(−3..−1)	T	PT(+1)	T

Dodatkowe atrybuty wyuczone na podstawie błędów

- Nie możemy wykorzystać formy ortograficznej dla wszystkich słów, ale dla małej liczby wyjątków gramatycznych możemy.
- Jak je znaleźć?
- Klasyfikator odnajduje reguły/regularności. Myli się na wyjątkach.
- Rozwiązanie:

wyjątek?(−3..+1)	WPL(−3..+1)	P(1)	S(1,2,3)	PT(−3..−1)	T	PT(+1)	T

Dodatkowe atrybuty wyuczone na podstawie błędów

- Nie możemy wykorzystać formy ortograficznej dla wszystkich słów, ale dla małej liczby wyjątków gramatycznych możemy.
- Jak je znaleźć?
- Klasyfikator odnajduje reguły/regularności. Myli się na wyjątkach.
- Rozwiązanie:

wyjątek?(−3..+1)	WPL(−3..+1)	P(1)	S(1,2,3)	PT(−3..−1)	T	PT(+1)	T

Dodatkowe atrybuty wyuczone na podstawie błędów

- Nie możemy wykorzystać formy ortograficznej dla wszystkich słów, ale dla małej liczby wyjątków gramatycznych możemy.
- Jak je znaleźć?
- Klasyfikator odnajduje reguły/regularności. Myli się na wyjątkach.
- Rozwiązanie:

wyjątek?(−3..+1)	WPL(−3..+1)	P(1)	S(1,2,3)	PT(−3..−1)	T	PT(+1)	T

Informacje z analizatora leksykalnego

- Opisywany dotąd algorytm zgaduje tagi dla wszystkich słów.
- Wszystkie tagi są dozwolone (przez tagset) ale mogą nie być poprawne dla konkretnego słowa.
- np. rodzaj dla rzeczowników – często kontekst dopuszcza różne rodzaje, ale forma ortograficzna ma jeden konkretny.
- Analizator morfologiczny (morfeusz) pozwala poprawić takie przypadki.

Informacje z analizatora leksykalnego

- Opisywany dotąd algorytm zgaduje tagi dla wszystkich słów.
- Wszystkie tagi są dozwolone (przez tagset) ale mogą nie być poprawne dla konkretnego słowa.
- np. rodzaj dla rzeczowników – często kontekst dopuszcza różne rodzaje, ale forma ortograficzna ma jeden konkretny.
- Analizator morfologiczny (morfeusz) pozwala poprawić takie przypadki.

Informacje z analizatora leksykalnego

- Opisywany dotąd algorytm zgaduje tagi dla wszystkich słów.
- Wszystkie tagi są dozwolone (przez tagset) ale mogą nie być poprawne dla konkretnego słowa.
- np. rodzaj dla rzeczowników – często kontekst dopuszcza różne rodzaje, ale forma ortograficzna ma jeden konkretny.
- Analizator morfologiczny (morfeusz) pozwala poprawić takie przypadki.

Poprawianie parametrów

- Wszystkie opisane usprawnienia wpływają na siebie wzajemnie.
- Ze względów wydajnościowych poszukiwanie sensownych wartości było przeprowadzone na małym korpusie rozwojowym.
- Ostateczne wartości dobrane na większym korpusie ze 120 tys. słów.
- Optymalne wartości się nieco zmieniły. . .

Poprawianie parametrów

- Wszystkie opisane usprawnienia wpływają na siebie wzajemnie.
- Ze względów wydajnościowych poszukiwanie sensownych wartości było przeprowadzone na małym korpusie rozwojowym.
- Ostateczne wartości dobrane na większym korpusie ze 120 tys. słów.
- Optymalne wartości się nieco zmieniły. . .

Poprawianie parametrów

- Wszystkie opisane usprawnienia wpływają na siebie wzajemnie.
- Ze względów wydajnościowych poszukiwanie sensownych wartości było przeprowadzone na małym korpusie rozwojowym.
- Ostateczne wartości dobrane na większym korpusie ze 120 tys. słów.
- Optymalne wartości się nieco zmieniły...

Ostateczne wartości parametrów

- Prawy i lewy kontekst: 1 słowo,
- osobny rozmiar kontekstu dla "wyjątków gramatycznych": 2 słowa dla lewego, 1 słowo dla prawego kontekstu,
- sufiksy formy ortograficznej: 1-, 2- i 3-literowy,
- bez prefiksów formy ort.,
- "wyjątki gramatyczne": 1-literowe prefiks i sufiks dla lematu i 2-literowy sufiks formy ortograficznej (tylko przetwarzane słowo).

wyj?(–2..+1)	wlP(1)	wlS(1)	wS(2)	WPL(–1..+1)	S(1,2,3)	PT(–1)	T	PT(+1)	T

Wyniki dla różnych kombinacji ulepszeń

			bez formy ort.			z formą ort.		
			T	PT k.+słowo	PT k. (tylko)	T	PT k.+słowo	PT k. (tylko)
w.−	a.m.−	k.−	91.04	90.89	91.04	94.43	94.45	94.43
		k.+	91.11	91.25	91.40	94.56	94.50	94.56
	a.m.+	k.−	92.79	92.62	92.79	94.74	94.74	94.74
		k.+	92.85	93.02	93.16	94.85	94.84	94.93
w.+	a.m.−	k.−	94.91	94.72	94.91	95.12	95.01	95.12
		k.+	94.99	94.99	95.03	95.16	95.10	95.13
	a.m.+	k.−	95.00	94.77	95.00	95.12	95.03	95.12
		k.+	95.06	95.08	95.17	95.14	95.11	95.18

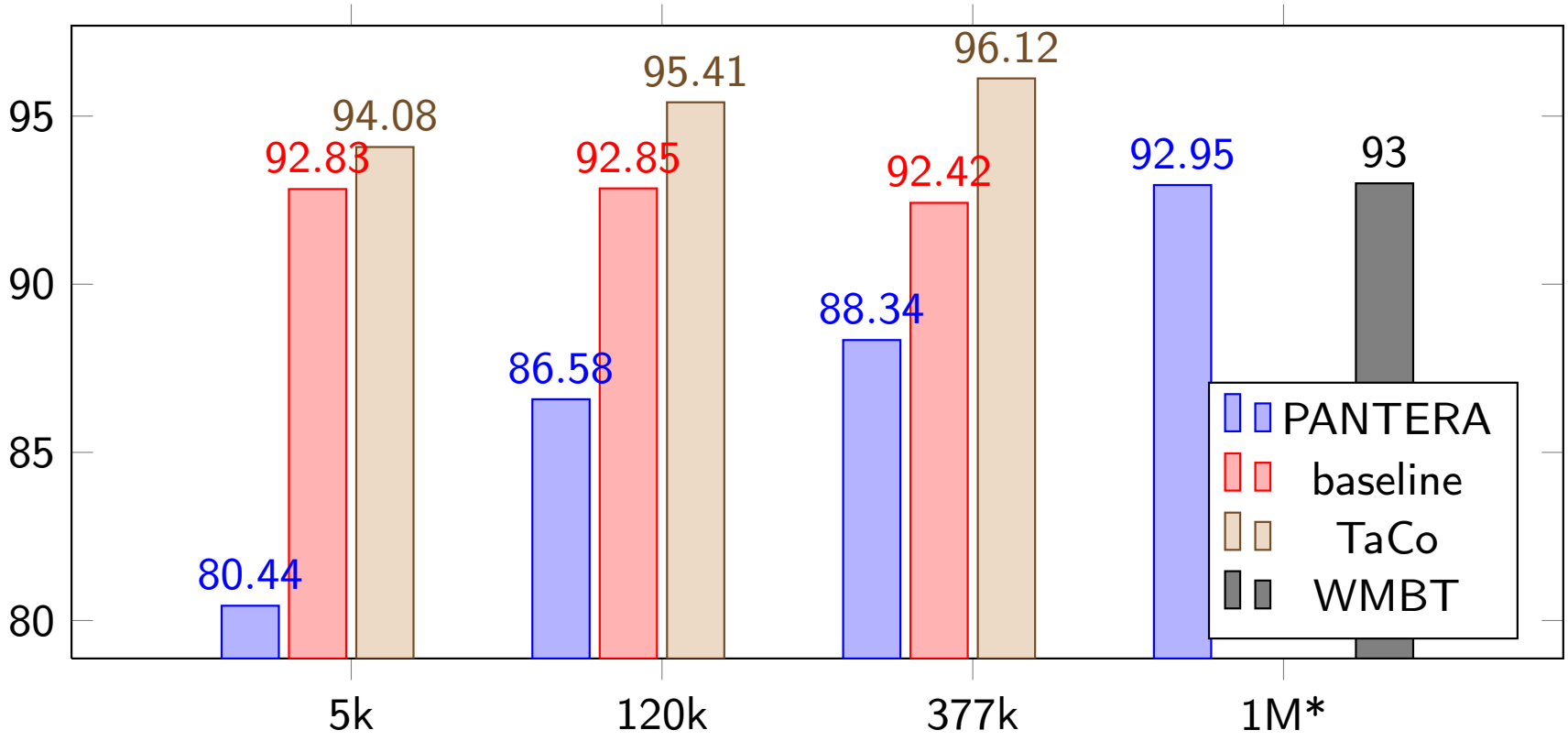
forma ort. – informacje z formy ortograficznej (WPL, prefiks, sufiks)
w. – informacje o wyjątkach
a.m. – analiza morfologiczna
k. – kontekst
T – tag (postać zwykła)
PT – tag kodowany pozycyjnie

Wyniki ewaluacji

- Korpus: część wzbogaconego korpusu Słownika frekwencyjnego polszczyzny współczesnej – anotowany ręcznie za pomocą tagsetów KIPI i NKJP.
- $\approx$ 377k słów, różne gatunki tekstów (popularnonaukowe, wiadomości prasowe, publicystyka, dramat).
- Korpus przeanalizowany za pomocą Morfeusza (56.1% słów niejednoznacznych, 1.5% nieznanych analizatorowi).
- Tagset KIPI: $\approx$ 1350 tagów, 913 użytych w anotacji,
- Tagset NKJP: $\approx$ 1700 tagów, 792 użytych w anotacji,
- Tagsety są podobne: konwersja 90.6% słów trywialna (automatyczna zamiana napisów).

Wyniki ewaluacji

- Korpus: część wzbogaconego korpusu Słownika frekwencyjnego polszczyzny współczesnej – anotowany ręcznie za pomocą tagsetów KIPI i NKJP.
- $\approx$ 377k słów, różne gatunki tekstów (popularnonaukowe, wiadomości prasowe, publicystyka, dramat).
- Korpus przeanalizowany za pomocą Morfeusza (56.1% słów niejednoznacznych, 1.5% nieznanych analizatorowi).
- Tagset KIPI: $\approx$ 1350 tagów, 913 użytych w anotacji,
- Tagset NKJP: $\approx$ 1700 tagów, 792 użytych w anotacji,
- Tagsety są podobne: konwersja 90.6% słów trywialna (automatyczna zamiana napisów).



algorytm	korpus	poprawność			czas, pamięć
		całość	niejednozn.	nietrywialne	
TaCo	377k	96.12%	93.08%	58.54%	20h, 10.5GB
baseline	377k	92.42%	86.49%	19.00%	5min, 600MB
TaCo	120k	95.41%	91.82%	50.95%	9h, 1.9GB
TaCo	5k	94.08%	89.45%	36.74%	6min, 80MB

Tło dla wyników – inne metody

- Klasyczne taggery – na wykresie (poprawność: 80%..93%).
- DZ Interset (Daniel Zeman):
  - reguły dla KIPI istnieją, dla NKJP brak,
  - poprawność szacunkowo ok. 91-92%.
- nieoficjalny(?) konwerter KIPI->NKJP (Michał Lenart i inni) – poprawność ok. 94%.

Analiza błędów

% wszystkich błędów	tag źródłowy	tag docelowy (gold standard)	tag wybrany
2.14	conj	qub	conj
1.91	adv:pos	qub	adv:pos
1.73	qub	conj	qub
1.54	qub	qub	conj
0.99	qub	adv	qub
0.88	adv:pos	adv:pos	adv
0.82	conj	conj	qub
0.63	adj:pl:gen:m3:pos	adj:pl:gen:n:pos	adj:pl:gen:m3:pos

Główne przyczyny błędów:

- niekonsekwentna anotacja manualna,
- atrybuty opcjonalne.

Podsumowanie

- Konwersja tagsetów może być ulepszona dzięki skorzystaniu z istniejącej anotacji.
- Najbardziej wartościowe informacje: forma ortograficzna i "wyjątki gramatyczne".
- W przeciwieństwie do klasycznych taggerów TaCo osiąga dobre wyniki nawet na małych danych treningowych.

Strona domowa TaCo: `http://zil.ipipan.waw.pl/TaCo`

Podsumowanie

- Konwersja tagsetów może być ulepszona dzięki skorzystaniu z istniejącej anotacji.
- Najbardziej wartościowe informacje: forma ortograficzna i "wyjątki gramatyczne".
- W przeciwieństwie do klasycznych taggerów TaCo osiąga dobre wyniki nawet na małych danych treningowych.

Strona domowa TaCo: `http://zil.ipipan.waw.pl/TaCo`