Integracja narzędzi do przetwarzania języka polskiego we frameworku spaCy

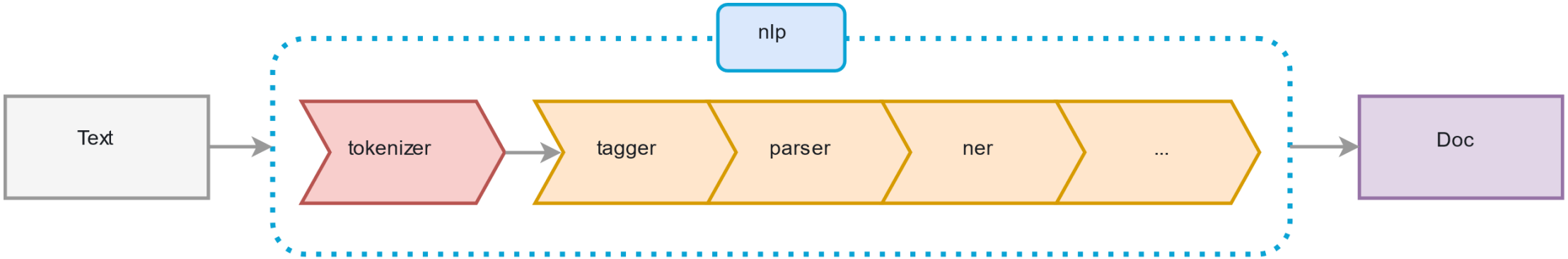
Ryszard Tuora, Łukasz Kobyliński
IPI PAN
Warszawa, 13 stycznia 2020

Plan referatu

1. Czym jest spaCy?
2. Implementacja spaCy dla języka polskiego
3. Wyniki i zastosowania
4. Dalsze kroki

spaCy

spaCy



spaCy to **open-source**'owa biblioteka **ogólnego zastosowania**, do NLP dla Pythona.

Twórcy spaCy celują w **łatwość użycia**, i **zastosowania produkcyjne**.

Standardowy pipeline składa się z **taggera**, **parsera** i komponentu **NER**.

Najnowsza wersja (2.2.3) została wypuszczona 21-go listopada.

spaCy - Popularność

	pattern	spaCy	nltk
Suma pobrań:	5,265,217	11,473,890	37,939,784
Ostatnie 30 dni:	324,977	751,746	2,486,742
Ostatnie 7 dni:	71,758	163,892	512,728

Źródło: https://pepy.tech

spaCy - Szybkość

	ABSOLUTE (MS PER DOC)			RELATIVE (TO SPACY)		
SYSTEM	TOKENIZE	TAG	PARSE	TOKENIZE	TAG	PARSE
spaCy	0.2ms	1ms	19ms	1x	1x	1x
CoreNLP	0.18ms	10ms	49ms	0.9x	10x	2.6x
ZPar	1ms	8ms	850ms	5x	8x	44.7x
NLTK	4ms	443ms	*n/a*	20x	443x	*n/a*

spaCy po polsku

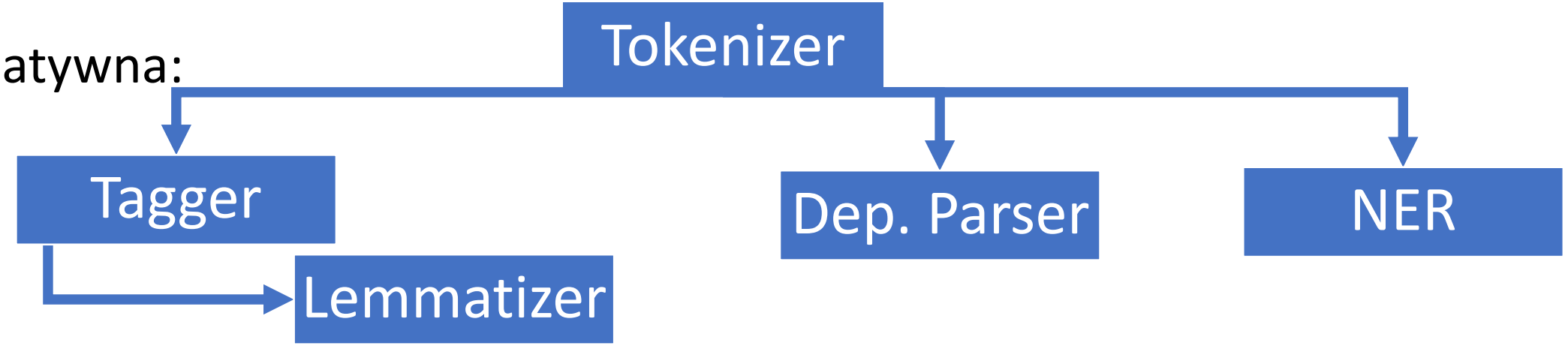
- 16 oficjalnych modeli dla 10 języków
- Brak oficjalnego modelu dla języka polskiego
- Istniejące rozwiązania i zasoby dla NLP języka polskiego (clip.ipipan.waw.pl/LRT)
- Brak alternatywnych pipeline'ów integrujących w prosty sposób zasoby dla polskiego

Stwórzmy **spaCy**-PL!

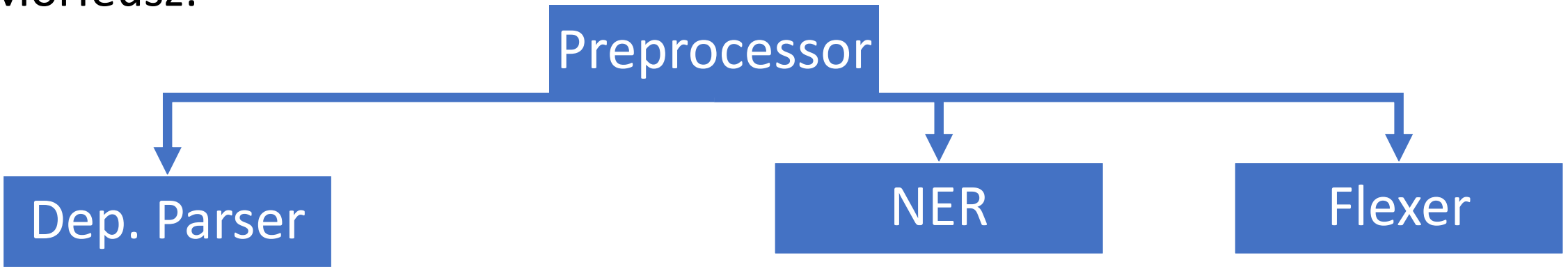
Implementacja

2 wersje spaCy-PL

Natywna:



Morfeusz:



2 wersje spaCy-PL

Natywna	**„morfeuszowa”**
- Szybsza	- Dokładniejsza
- Samodzielna	- Wymaga instalacji Morfeusza i kilku modułów
- Tylko tagi POS	- Pełna informacja morfosyntaktyczna
-	- Segmentacja aglutynantów

Wersja natywna

spaCy – wersja natywna

- Embeddingi KGR10, 100 wymiarów obcięte do 250 tys. najczęściej pojawiających się słów.
- Natywny tokenizator oparty na domyślnych regułach
- Natywny tagger
- Przygotowany przez nas, słownikowy lematyzator
- Natywny parser
- Natywny komponent NER
- 166 MB po spakowaniu

spaCy – Architektura

W komponentach natywnych, spaCy wykorzystuje dynamiczną reprezentację wektorową tokenów.

W budowaniu tej reprezentacji bierze pod uwagę zarówno embeddingi, jak i np. suffiksy, oraz kształt napisu. Konstruuje ją osobna sieć konwolucyjna.

Reprezentacja jest potem wykorzystywana w neuronowym klasyfikatorze.

spaCy - Tagger

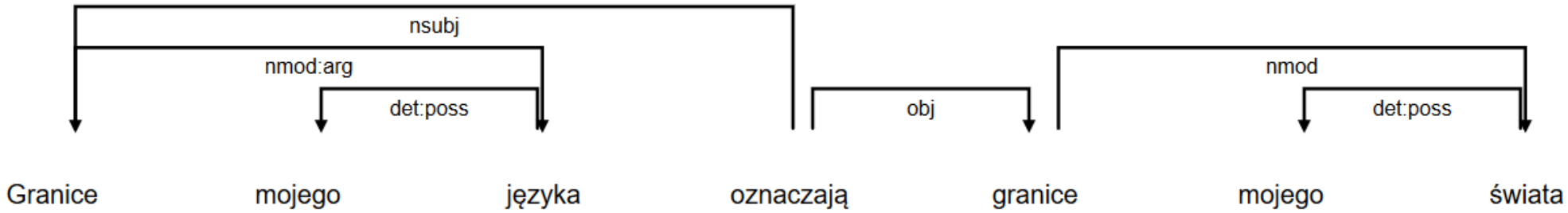
- Wytrenowany na NKJP + Korpusie frekwencyjnym z lat 60tych.
- Korzysta z tagsetu NKJP
- Rzutuje także (w prosty sposób) wyniki na tagset UD.
- Nie wykonuje pełnej analizy morfosyntaktycznej

NCP tag	UD tag	NCP tag	UD tag
ADJ	ADJ	INTERJ	INTJ
ADJA	ADJ	INTERP	PUNCT
ADJA	ADJ	NUM	NUM
ADJC	ADJ	NUMCOL	NUM
ADJP	ADJ	PACT	VERB
ADV	ADV	PANT	VERB
AGLT	AUX	PCON	VERB
BEDZIE	VERB	PPAS	VERB
BREV	X	PPRON12	PRON
BURK	ADV	PPRON3	PRON
COMP	SCONJ	PRAET	VERB
CONJ	CCONJ	PRED	VERB
DEPR	NOUN	PREP	ADP
FIN	VERB	QUB	PART
GER	NOUN	SIEBIE	PRON
IMPS	VERB	SUBST	NOUN
IMPT	VERB	WINIEN	VERB
INF	VERB	XXX	X

spaCy - Lematyzator

- Przygotowany przez nas komponent, wykorzystujący wprowadzone w 2.2 implementację lookup-tables.
- Słownik lematów SGJP pochodzi z Morfeusza, i został podzielony na 10 podsłowników forma -> lemat ze względu na tag UD.
- Wykorzystuje kilka trików pozwalających na przyspieszenie i redukcję zajmowanego miejsca (np. odcinanie prefiksu negacji „nie-”)

spaCy - Parser zależnościowy



- Wytrenowany na korpusie PDB
- Wykorzystuje jedynie reprezentację wektorową słów
- Dzieli dokumenty na zdania (wymaga to sztucznego tworzenia wielozdaniowych dokumentów)
- Oparty o metodologię transition-based
- spaCy umożliwia łatwą wizualizację drzew zależnościowych

spaCy - komponent NER

- Wytrenowany na NKJP w podstawowej wersji zbioru oznaczeń
- Wykorzystuje jedynie reprezentację wektorową słów
- Rozpoznaje 6 typów obiektów: persName, orgName, geogName, placeName, date, time
- spaCy nie zezwala na obiekty zagnieżdżone (e.g. [Plac [Piłsudskiego]], lub [al. [3 maja]])
- Implementuje metodologię transition based

Wersja „morfeuszowa”

spaCy – wersja „morfeuszowa”

- Embeddingi KGR10, 100 wymiarów, obcięte do 250 tys. najczęściej pojawiających się słów.
- Przygotowany przez nas „Preprocessor” spełniający zadania tokenizatora, taggera morfosyntaktycznego, i lematyzatora.
- Natywny parser
- Natywny komponent NER
- Przygotowany przez nas komponent do fleksji („flexer”)
- 164 MB po spakowaniu (nie wliczając Morfeusza i wykorzystywanych modułów)

spaCy - Preprocessor: segmentacja

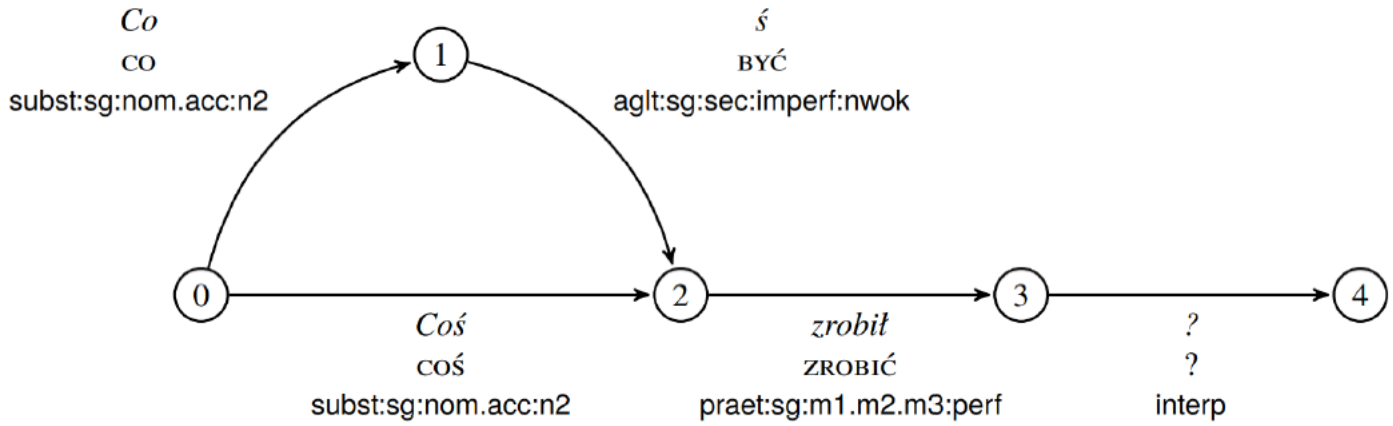


Figure 1: Ambiguous segmentation as analyzed by Morfeusz analyzer (Woliński, 2014).

- Do segmentacji wykorzystujemy Morfeusza 2, dzięki temu uzyskujemy bogatszą listę wyjątków segmentacyjnych, oraz wyodrębnienie aglutynantów, które zbliża model do metodologii przyjętej w wykorzystywanych korpusach.
- Z wyjątkiem słów [„coś”, „ktoś”, „kogoś”, „gdzieś”, „kiedyś”] zawsze wybieramy dłuższą ścieżkę w grafie.

spaCy - Preprocessor: tagowanie, lematyzacja

- Do tagowania **pełnymi tagami** wykorzystujemy Morfeusza 2, wraz z Toyggerem jako dezambiguatorem

- Tagi POS przechowujemy w atrybucie `token.tag_`, zaś cechy morfosyntaktyczne, w dodanym atrybucie `token._.feats`.

- Dezambiguacja pozwala na wybranie właściwego lematu spośród hipotez Morfeusza.

spaCy - Preprocessor: integracja Toyggera

- Wytrenowaliśmy model na wykorzystanych przez nas embeddingach.

- Zbiór treningowy jest taki sam jak w wypadku natywnego taggera.

Toygger:

I. dostaje na wejściu propozycje tagów od Morfeusza, oraz reprezentację wektorową dokumentu.
II. generuje własny tag
III. wybiera najbliższy w odległości Levenshteina tag z zaproponowanych przez Morfeusza, i związany z nim lemat
IV. rzutuje tag w zbiór tagów NKJP.
V. zapisuje tag, lemat i cechy morfosyntaktyczne w tokenie

- Jeżeli Morfeusz nie rozpoznaje danego słowa, zapisujemy tag zwrócony przez Toygger.

spaCy - „Flexer”

- Przygotowany przez nas komponent wykorzystujący Morfeusza.
- Dla podanego tokenu zwraca formę o pożądanej charakterystyce morfosyntaktycznej (najbliższą do charakterystyki wyjściowej).
- Np. `flexer.flex(„pies”, „gen:pl”) -> „psów”`
  `flexer.flex(„zjadła”, „m1”) -> „zjadł”`

- Flexer:

I. używa Morfeusza do wygenerowania innych form danego tokenu
II. zawęża ten zbiór do form spełniających żądane własności
III. wybiera formę najbliższą w odległości Levenshteina do charakterystyki tokenu wyjściowego.

Wykorzystanie

Instalacja

- spaCy instaluje się przez PIP
- Nasze modele, również instalowane przez PIP, są dostępne tutaj: zil.ipipan.waw.pl/SpacyPL
- Więcej informacji można znaleźć tu: github.com/ipipan/spacy-pl

Wykorzystanie spaCy-PL

```
>>> import pandas # for visualization
>>> import spacy
>>> nlp = spacy.load("pl_spacy_model")

# Tokenization, Tagging, Lemmatization and Dependency Parsing
>>> sent1 = "Granice mojego języka oznaczają granice mojego świata" # ~Wittgenstein
>>> doc1 = nlp(sent1)
>>> attribs = ['orth_', 'lemma_', 'tag_', 'pos_', 'dep_', 'head']
>>> table = [{att:tok.__getattribute__(att) for att in attribs} for tok in doc1]
>>> df = pandas.DataFrame(table)
>>> print(df[attribs])
```

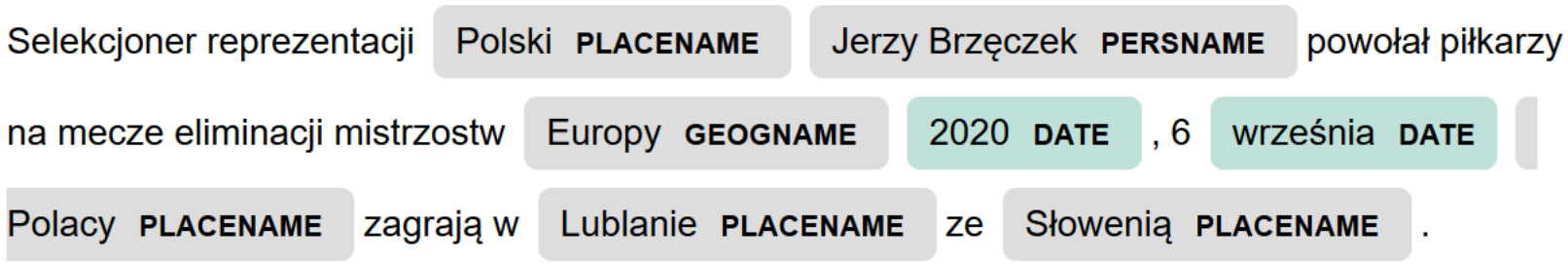
	orth_	lemma_	tag_	pos_	dep_	head
0	Granice	granica	SUBST	NOUN	nsubj	oznaczają
1	mojego	mój	ADJ	ADJ	det:poss	języka
2	języka	język	SUBST	NOUN	nmod:arg	Granice
3	oznaczają	oznaczać	FIN	VERB	ROOT	oznaczają
4	granice	granica	SUBST	NOUN	obj	oznaczają
5	mojego	mój	ADJ	ADJ	det:poss	świata
6	świata	świat	SUBST	NOUN	nmod	granice

NER w spaCy-PL

```
# NER
>>> sent2 = "Selekcjoner reprezentacji Polski Jerzy Brzęczek \
        powołał piłkarzy na mecze eliminacji mistrzostw Europy 2020,\
        6 września Polacy zagrają w Lublanie ze Słowenią." # ~Rzeczpospolita
>>> doc2 = nlp(sent2)

# NER visualization

>>> displacy.render(doc2, style="ent")
```



Chatboty z wykorzystaniem spaCy-PL i RASA

RASA to open-source'owy framework do tworzenia agentów konwersacyjnych.

RASA jest blisko związana ze spaCy, i umożliwia bezpośrednie wykorzystanie modeli spaCy do przygotowania podstawowych funkcjonalności chatbota.

Obecnie komponenty najważniejsze dla RASY to embeddingi, tokenizator, lematyzator i NER.

Chatboty z wykorzystaniem spaCy-PL i RASA

```
 2                                                                    Cześć !
                                                          intent: greet 0.40
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
 3    utter_greet 1.00
      Cześć, jak mogę Ci pomóc?
      action_listen
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
 4                                    Podaj mi numer komórkowy prezydenta
[Andrzeja](employee_first_name:andrzeja) [Dudy](employee_surname:dudy)
                                                       intent: ask_phone 0.98
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
 5    slot{"employee_first_name": "andrzeja"}
      slot{"employee_surname": "dudy"}
      action_provide_phone 0.80
      Telefon andrzeja to A
      action_listen 1.00
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
 6                                                  Dzięki , do usłyszenia !
                                                          intent: goodbye 0.60
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
 7    utter_goodbye 0.93
      Dzięki za rozmowę! Do zobaczenia wkrótce!
```

Ewaluacja

Zbiory treningowe

Komponent	Zbiór treningowy
Tagger:	NKJP + KF 60’s
Parser	PDB
NER	NKJP bez 500 dokumentów wykorzystanych potem jako zbiór testowy Etykiety obcięte do 6 podstawowych typów Zarówno ze zbioru testowego, jak i treningowego usunięto zagnieżdżone jednostki

Ewaluacja

	name	Sentences f1	Words f1	Tokens f1	XPOS f1	Feats f1	UPOS f1	Lemmas f1	UAS f1	LAS f1
Morfeusz	pl_lfg_test	99.8%	99.8%	99.8%	95.3%	92.3%	83.3%	97.8%	91.3%	86.1%
	pl_pdb_test	99.0%	99.8%	98.5%	92.0%	84.6%	84.1%	97.4%	91.5%	88.6%
	pl_pud_test	95.9%	99.7%	99.3%	89.3%	80.9%	79.0%	96.4%	90.1%	86.1%
Standard	pl_lfg_test	99.8%	96.7%	96.7%	96.1%		83.3%	91.0%	84.6%	79.6%
	pl_pdb_test	98.5%	97.6%	98.8%	90.8%		83.9%	91.1%	86.6%	83.8%
	pl_pud_test	96.4%	97.8%	98.2%	90.9%		78.5%	87.8%	86.1%	82.8%

Każdy z uprzednio wytrenowanych (w opisany wyżej sposób) **dwu** modeli był przetestowany na zbiorze testowym każdego z trzech polskich korpusów UD (LFG, PDB, PUD).

Wyniki uzyskano przy pomocy skryptu ewaluacyjnego conll18

Ewaluacja

	total	orgName	geogName	placeName	date	persName	time
precision	85.9%	82.6%	71.7%	84.9%	91.1%	90.8%	57.9%
recall	86.4%	81.6%	72.4%	85.1%	90.2%	91.5%	84.6%
f1	86.1%	82.1%	72.1%	85.0%	90.6%	91.1%	68.8%

Ewaluacja komponentu NER jest trudniejsza, ponieważ nie spełnia on założeń np. danych ewaluacyjnych z PolEval 2018.

Powyższe wyniki uzyskujemy na wyodrębnionym wcześniej testowym podzbiorze 500 dokumentów z NKJP, przy zastosowaniu skryptów ewaluacyjnych od spaCy. Skrypty te zaliczają tylko dokładne pokrycie obiektów.

Wyniki charakteryzują się dużą wariancją między klasami tagów.

Plany na przyszłość:

- Optymalizacja Toyggera
- Przygotowanie modeli o różnych rozmiarach, (np. modelu wykorzystującego wektory 300-wymiarowe, lub mniejszego modelu do zastosowań edukacyjnych)
- Dodatkowe komponenty pipeline'u (np. analiza wydźwięku i emocji, autokorekta)

Literatura:

- Dziubalska-Kolaczyk, K. et al., editors. (2019). Current state of the art in language technology for Polish, special issue of Poznan Studies in Contemporary Linguistics, volume 55(2). De Gruyter.
- Honnibal, M., & Johnson, M. *(2015).* An Improved Non-monotonic Transition System for Dependency Parsing. *EMNLP.*
- Kocoń, J. and Gawor, M. (2018). Evaluating KGR10 Polish word embeddings in the recognition of temporal expressions using BiLSTM-CRF. Schedae Informaticae, 27.
- Krasnowska-Kieraś, K. (2017). Morphosyntactic disambiguation for Polish with bi-LSTM neural networks. In Zygmunt Vetulani et al., editors, Proceedings of the 8th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics, pages 367–371, Pozna´n, Poland. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu.
- Przepiórkowski, A., et al., editors. (2012). Narodowy Korpus Jezyka Polskiego. Wydawnictwo Naukowe PWN, Warsaw.
- Tuora, R., and Kobyliński, Ł. (2019) Integrating Polish language tools and resources in Spacy. In *Proceedings of PP-RAI 2019 Conference*, pages 210–214, Wrocław, 2019. Department of Systems and Computer Networks, Faculty of Electronics, Wroclaw University of Science and Technology.
- Woliński, M. (2014). Morfeusz reloaded. In Nicoletta Calzolari, et al., editors, Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, pages 1106–1111, Reykjavik, Iceland. ELRA.

Dziękujemy za uwagę!